AD-A094 353 ANALYTICS INC WILLOW GROVE PA F/G 5/8
THE HUMAN OPERATOR SIMULATOR. VOLUME IX. HOS STUDY GUIDE.(U)
SEP 78 M I STRIEB, F A GLENN, R J WHERRY N62269-78-M-6684
UNCLASSIFIED TR-1320-VOL-9 NL

1 of 4
AD A
094353

2

Technical Report 1320

THE HUMAN OPERATOR SIMULATOR
VOLUME IX
HOS STUDY GUIDE

Contract No. N62269-78-M-6684

September, 1978

Prepared by:
Melvin I. Strieb
Floyd A. Glenn, PhD
Robert J. Wherry, Jr., PhD

ANALYTICS
2500 MARYLAND ROAD, WILLOW GROVE, PA. 19090

14 TR-1320-VOL-9

9 Technical Report 1320

6

THE HUMAN OPERATOR SIMULATOR.
VOLUME IX.
HOS STUDY GUIDE.

15

Contract No. N62269-78-M-6684

11 Septembe[redacted]978

12 310

Prepared by:

10

Melvin I. Strieb
Floyd A. Glenn, [redacted]
Robert J. Wherry, Jr. [redacted]

ANALYTICS
2500 MARYLAND ROAD, WILLOW GROVE, PA 19090

405904 JM

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 HOS IS A SOLUTION TO A SYSTEM EVALUATION PROBLEM

In recent years, as "systems" (whether spacecraft, jet fighters, or office equipment) have become more complex, there has been a growing concern about the extent to which these systems are adapted to the capabilities of the individuals who are called on to operate them. One of the primary emphases in the field of human factors engineering has been on developing methodologies to *evaluate* systems to ensure that they are as well adapted as possible to human capabilities and limitations. A major thrust in these efforts has been to develop methods that would enable the operability of a system to be evaluated *before* the system is ever built. Typically, this has been done by building prototypes or simulators which are "flown" with an operator "in-the-loop." The operator's performance is then measured and judged against a set of performance criteria. The measures, along with the operator's comments are used intuitively to "select" changes that are intended to improve system performance. Thus, the methodology currently used to evaluate systems -- building prototypes or testing designs in simulators -- relies on the construction of *real* hardware that can be tested with *real* operators. This requirement inevitably means that there is a substantial time lag between the system's initial design and the time when the prototype can be built, tested, evaluated, and any proposed changes "recycled" through the system design process. In addition, the use of dynamic simulators that use real operators inevitably requires a substantial outlay of money for the development of attendant hardware and software, for the training and retraining of operators, and for the performance, analysis, and evaluation of controlled experiments that utilize the simulators. Consequently, while dynamic simulators can be highly useful in *training* operators, it is doubtful that the data collected from them has a significant impact on system design.

HOS WAS DEVELOPED TO ENABLE SYSTEMS TO BE EVALUATED

- EARLY IN DESIGN PROCESS
- WITH MAN-IN-THE-LOOP
- WITHOUT HAVING TO BUILD HARDWARE OR SOFTWARE
- IN A WAY THAT WOULD ENSURE CONSISTENT, COMPARABLE, AND REPRODUCIBLE RESULTS

The Human Operator Simulator (HOS) was developed to alleviate these problems in two ways. First, by enabling potential problem areas to be identified earlier in the system design process, HOS enables corrections to be made at lower cost and with less disruption of development schedules. Second, for areas of marginal operator performance, HOS can focus dynamic simulations on the likely sources of difficulty, reducing the time and cost required to validate system operability.

As soon as the definition of the functional requirements and the allocation of functions are completed, a HOS simulation can be developed that will describe both the hardware and the procedures that the operator is to use in order to control a proposed piece of equipment. These descriptions are then combined with a "problem description" that defines the operator's environment and HOS then simulates the system as if it were a *real* operator using a *real* piece of equipment with a *real* job to do and a *real* problem to solve.

Unlike other "operator simulators" that simply draw times for task executions from sample distributions, HOS is a *detailed model of an operator*. The HOS operator has *eyes, hands,* and *feet* that *move*, absorb information from *displays*, and manipulate *controls* in accordance with the analyst's instructions. HOS has a *mind* that can perform mental calculations that *remembers* and *forgets* information. The HOS operator will perform the actions that are necessary in order to accomplish a task, but will omit actions that may be unnecessary because of the situation in which the operator finds himself.

1.2 WHAT INFORMATION CAN BE OBTAINED BY SIMULATING AN OPERATOR?

First, HOS enables an analyst to perform operator-in-the loop "experiments" that produce consistent, comparable, and reproducible results. This is important because one of the major problems that plagues human performance experimentation is the difficulty of providing proper controls on

ADVANTAGES OF SIMULATION

-- OPERATOR CHARACTERISTICS AND TASKS CAN BE CONTROLLED

-- ENVIRONMENTAL CONDITIONS AND PROBLEM SITUATIONS CAN BE CONTROLLED

-- PROCESSES CAN BE OBSERVED, MEASURED, AND RECORDED WITHOUT INTERFERRING WITH PERFORMANCE all the potentially relevant experimental factors. Variability in the ways in which operators may choose to carry out tasks in a particular situation often makes it difficult to compare the results obtained on different experimental trials. Clearly, there is much information to be learned by observing the full range of performance. However, in order to be able to ensure a fair evaluation of a proposed system, one would prefer to have consistent average performance. HOS gives the analyst the ability to experiment with an operator whose performance is guaranteed to be consistent and typical.

At the same time, HOS permits experimentation to be performed with operators who deviate from the average in a variety of ways. The HOS operator is, theoretically, a *trained operator* of *average capabilities* who will carry out his instructions exactly as he is told to do, exhibiting some variability in performance speed, but, in the long run, taking the amount of time for each instruction that an average operator would have taken. However, by suitably changing some of the parameters, equations, or instructions in the simulation, an analyst can change the HOS operator into a highly idiosyncratic individual.

The second advantage that is gained by having the ability to simulate a human operator is that one can readily simulate different environmental conditions to examine how the operator responds to changing situations. For example, one HOS simulation (Ref. 1) examined the performance of the Sensor Station 3 operator on board the P-3C ASW Patrol Aircraft during a simulated anchorage mission in the Mediterranean. Because it was a simulation, all the features of the problem environment could be controlled -- the numbers and types of targets, their locations and characteristics, emitter duty cycles, tactics, etc. The effects on the operator's performance when more targets were added to the search area could be readily examined and the conditions under which the operator became overloaded could be determined. In another simulation (Ref. 2), the HOS operator performed a tracking task and an interfering secondary task simultaneously. The effects of varying the characteristics of the signal that the operator was tracking and the frequency and duration of the interrupts caused by the secondary task could be studied under more precisely controlled conditions than would be achievable in the laboratory.

A third important advantage of simulation over live experimentation lies in the fact that a simulated process can be observed, measured, and recorded with perfect fidelity without any danger of having the observation process disturb the performance being observed. In live experiments, the performance of human subjects is always affected by the human's awareness that he is being observed. In some cases, the observation and measurement apparatus may actually restrict the subject's sensory and motor capabilities. In all cases, the experimental subject who knows that he is participating in an experiment will behave more cautiously and deliberately than a person who does not believe that he is being studied. A simulated human operator, on the other hand, will never modify his performance because he suspects that someone is watching.

1.3 CHARACTERISTICS OF THE HOS OPERATOR

The HOS operator is a *trained* operator -- i.e., an operator who is familiar with the equipment (i.e., knows the locations of all the displays and controls and their characteristics), and how to operate it (i.e., knows the procedures and mental calculations that must be performed). Since the operator is assumed to be trained, the operator will carry out the procedures and use the equipment unerringly. The operator will never perform an instruction out of sequence or other than instructed.

This does *not* mean, however, that the operator cannot make a mistake. There are at least two sources of error in the HOS operator's performance -- his short-term memory and his perceptual processes.* When

*There is actually a third source -- the operator could have been trained incorrectly, causing him to do things in the wrong sequence.

THE HOS OPERATOR

- A TRAINED OPERATOR

  -- KNOWS LOCATIONS OF DISPLAYS AND CONTROLS
  -- KNOWS PROCEDURES
  -- KNOWS MENTAL CALCULATION PROCESSES
  -- WILL NOT PERFORM INSTRUCTIONS OUT OF SEQUENCE OR OTHER THAN INSTRUCTED

- ERRORS ARE THE RESULT OF OPERATOR/EQUIPMENT LIMITATIONS

  -- "FALLIBLE" SHORT-TERM MEMORY
  -- PERCEPTUAL/MOTOR PROCESSES CAN CAUSE ERRORS IF DISPLAYS ARE HARD TO READ OR CONTROLS ARE HARD TO MANIPULATE

HOS REQUIRES DATA ON

- DISPLAY AND CONTROL LOCATIONS
- DISPLAY AND CONTROL CHARACTERISTICS
- HOW EACH DISPLAY AND CONTROL IS USED
- OPERATOR'S MISSION
- OPERATOR CHARACTERISTICS
- SPECIFIC PROBLEM ENVIRONMENT
- ENVIRONMENTAL/SYSTEM DYNAMICS

the operator is told (or decides) to read a display, for example, he may read it incorrectly, either because of some characteristic of the display or because he did not spend a sufficient amount of time reading the display. When attempting to remember a value, the operator might remember the value incorrectly. These errors can cascade down through the restof the simulation. But, as modeled by HOS, they represent realistic possibilities that might occur if a real operator were performing the same job with the same equipment. We will be saying more about the sources of operator error when we discuss the individual performance models in HOS.

1.4 WHAT INFORMATION DOES HOS REQUIRE?

Suppose that we wish to simulate a pilot in a proposed aircraft cockpit in order to evaluate the cockpit design. HOS would need a detailed description of:

- The location of each display and control in the proposed cockpit.
- The characteristics of each display and control.
- How the pilot uses each display and control to fly the aircraft -- e.g., how the displays and controls are used when taxiing, taking off, climbing, etc.
- Specific environmental conditions at the beginning of the problem -- e.g., the plane's altitude, airspeed, windspeed, etc.
- The pilot's task (mission) -- e.g., land the plane.
- The aircraft's flight characteristics.

HOS can then simulate both the performance of the operator and the aircraft as the operator carries out his job. Some of his information can be omitted or treated as a constant value; how this works will be explained later.

A HOS SIMULATION REQUIRES DESCRIPTIONS OF

- HOW THE SYSTEM IS CONFIGURED
- HOW THE SYSTEM FUNCTIONS
- HOW THE OPERATOR USES IT

  -- OPERATOR ACTIONS
  -- OPERATOR DECISIONS

Clearly, there is a lot of information that must be supplied to HOS. Preparing this information is not, admittedly, an easy job. When creating a HOS simulation, one generally has to search through many references accumulating scattered information about how each system component functions until a coherent pattern emerges as to how the system, as a whole, is configured, how it functions, and how the operator uses it. It is this last point -- how the operator uses the system -- that tends to be the stickiest point to tease out from available references. Typically, most systems operations manuals will describe what the operator must do in order to *perform a function,* but do not describe what must be done in order to *solve a problem.* For example, Naval operations manuals discuss in detail what an ASW acoustic sensor station operator must do in order to *enter* a fix that he obtains from his sonobuoys, but they do not describe the *decisions* that the operator must make, almost intuitively, in order to *obtain* the fix in the first place. These decisions include such things as deciding how many sensors to place, where and when to place them, what channels to listen to and when to listen, etc. These types of decisions are rarely elaborated upon in any standard manuals, but they are the sorts of decisions that a real operator must make. Clearly, in a system that is still on the drawing boards, this information is even harder to obtain. But teasing this information out of system designers is one of the real benefits that we feel HOS provides -- HOS forces system designers to think about what they are asking an operator to do and why and it requires that they put these operator task specifications down on paper. These *operator specifications* are just as important as hardware and software specifications to the overall specification of the system, but have all too frequently been ignored in the past.

1.5 HOS IS A COMPOSITE OF OPERATOR PERFORMANCE MODELS

In this guide, we will be demonstrating how one develops these operator task specifications for a simple problem and how the specifications are converted to the instructions that HOS is to follow. But, first, there is a more basic question that we must answer -- how will HOS actually carry out these instructions, i.e., what is the underlying model of the human operator that is incorporated into HOS?

HOS IS A COMPOSITE OF OPERATOR FERFORMANCE DERIVED FROM SELECTED HUMAN PERFORMANCE LITERATURE.

MODELS ARE ABLE TO BE CHANGED READILY WITH ADVANCES IN THE STATE-OF-THE-ART.

HOS IS BEING VALIDATED BY COMPARING RESULTS WITH OBSERVED PERFORMANCE.

The answer to this question is that HOS is, in fact, a composite of a number of different models of human performance.* The HOS program has been written in such a way that any one of these models can be changed fairly easily, as the state-of-the-art of human performance modeling changes. In this course, we will be describing the performance models as they currently exist in the program and, in addition, we will indicate what studies these models have been derived from. You may find that you disagree with the appropriateness of a particular model in particular situations or the relevance of the experiments on which the model is based to the problems to which it is being applied. Feel free to do so. We have simply selected a set of models and experiments that we felt best represents the current state of operator performance modeling for the types of problems we are dealing with. HOS can be adapted to accommodate other alternative formulations if these seem more appropriate. There were many situations in which sifficient data were unavailable to ensure the validity of a given model, as formulated in HOS. Thus, we cannot be certain that all the submodels in HOS are as valid as we hope they are. Rather than waiting until sufficient data might be collected or until others formulate models which could be generally accepted, we chose instead to get HOS into operation and then adapt and improve it as it becomes necessary, by comparing its results with observations of the performance of real human operators.

1.6 HOS PRIMITIVE FUNCTIONS

HOS considers the operator to be capable of performing seven primitive functions:

(1) Obtaining information.

(2) Remembering information.

*Note that the analyst's description of the procedures the operator must follow is also a model -- a model of how the analyst believes that the equipment should be operated and/or a model of how an operator will, in fact, use the equipment.

HOS PRIMITIVE FUNCTIONS

- INFORMATION ABSORPTION
- INFORMATION RECALL
- MENTAL COMPUTATION
- DECISION-MAKING
- ANATOMY MOVEMENT
- CONTROL-MANIPULATION
- RELAXATION

(3) Performing a mental computation.

(4) Making a decision.

(5) Moving a body part.

(6) Performing a control manipulation.

(7) Relaxing.

Every action that the HOS operator performs is a combination of one or more of these primitive functions. Although an analyst can write operator procedures that will force the operator to perform a particular primitive at a particular point in a sequence of actions,* generally the analyst will let HOS determine the primitives required to accomplish a particular task for itself.

The primitive functions are often either imbedded in, or contain within themselves, human performance models. For example, when a situation arises in which the operator must move his hand to a particular device, there is logic that determines which hand he will use. Similarly, when the operator attempts to recall some item of information, there is a *recall model* that is automatically assessed by the program that simulates the operator's short-term memory processes. Because of the level at which these models operate, we often refer to them as *micro-models*.

1.7 THE HUMAN OPERATOR PROCEDURES (HOPROC) LANGUAGE

The language that the analyst uses to describe the tasks the operator is to perform is called HOPROC, the Human Operator Procedures Language. HOPROC enables the analyst to access the micro-models directly

*Note that we did not say "at a particular point in time." The analyst has some control over the time at which the operator will perform each action, but not much. HOS itself determines how long each action will take and hence the time at which a particular action will take place.

THE HUMAN OPERATOR PROCEDURES (HOPROC) LANGUAGE

-- AN ENGLISH-LIKE LANGUAGE USED TO DESCRIBE BOTH THE OPERATOR'S TASKS AND THE FUNCTIONING OF THE HARDWARE

-- ALLOWS BOTH IMPLICIT AND EXPLICIT ACCESS TO THE HOS MICRO-MODELS

-- HOPROC INSTRUCTIONS LOOK LIKE THE INSTRUCTIONS THAT WOULD BE GIVEN TO REAL OPERATORS

-- CAN BE THE BASIS FOR SYSTEMS DOCUMENTATION AND TRAINING to describe how the operator is to perform his tasks. More commonly, however, the analyst simply describes through HOPROC the tasks that the operator is to perform. HOS itself will then determine which micro-models are needed in order to accomplish specific functions. Thus, HOS is like a real operator -- given a set of instructions (in HOPROC), it can determine for itself what actions are required in order to carry out the instructions. As we shall show in the next sections, the HOPROC instructions themselves look very much like the instructions that would be given to a real operator. Thus, the HOPROC procedures developed for an operator station could serve as the basis for the materials to train operators in the use of the system. In addition, portions of HOPROC describe the functions of the hardware and software in the crewstation. Thus, in addition to documenting operator functions within the crewstation, the HOPROC description of a crewstation provides complete documentation on the crewstation itself.

1.8 A SAMPLE HOS SIMULATION

The sample simulation that we will be using in this course is an analysis of one of the tasks performed by the P-3C Sensor Station 3 (SS-3) operator. Specifically, we will develop a simplified version of the operator's radar plotting procedures. In developing the simulation, we will describe all the major HOPROC language constructs and how one combines them into a description of the operator's tasks. Then we will run the problem through HOS, examine the outputs obtained at each phase, and discuss what can be learned from them about the operator's performance.

By way of an introduction, the following sections describe the SS-3 crewstation, the functions of the SS-3 controls, and the procedures that the operator must follow when plotting radar targets. These descriptions are then followed by examples of the outputs obtained by running HOS for a specific set of targets. Later in this guide, we will discuss both the inputs and outputs in more detail.

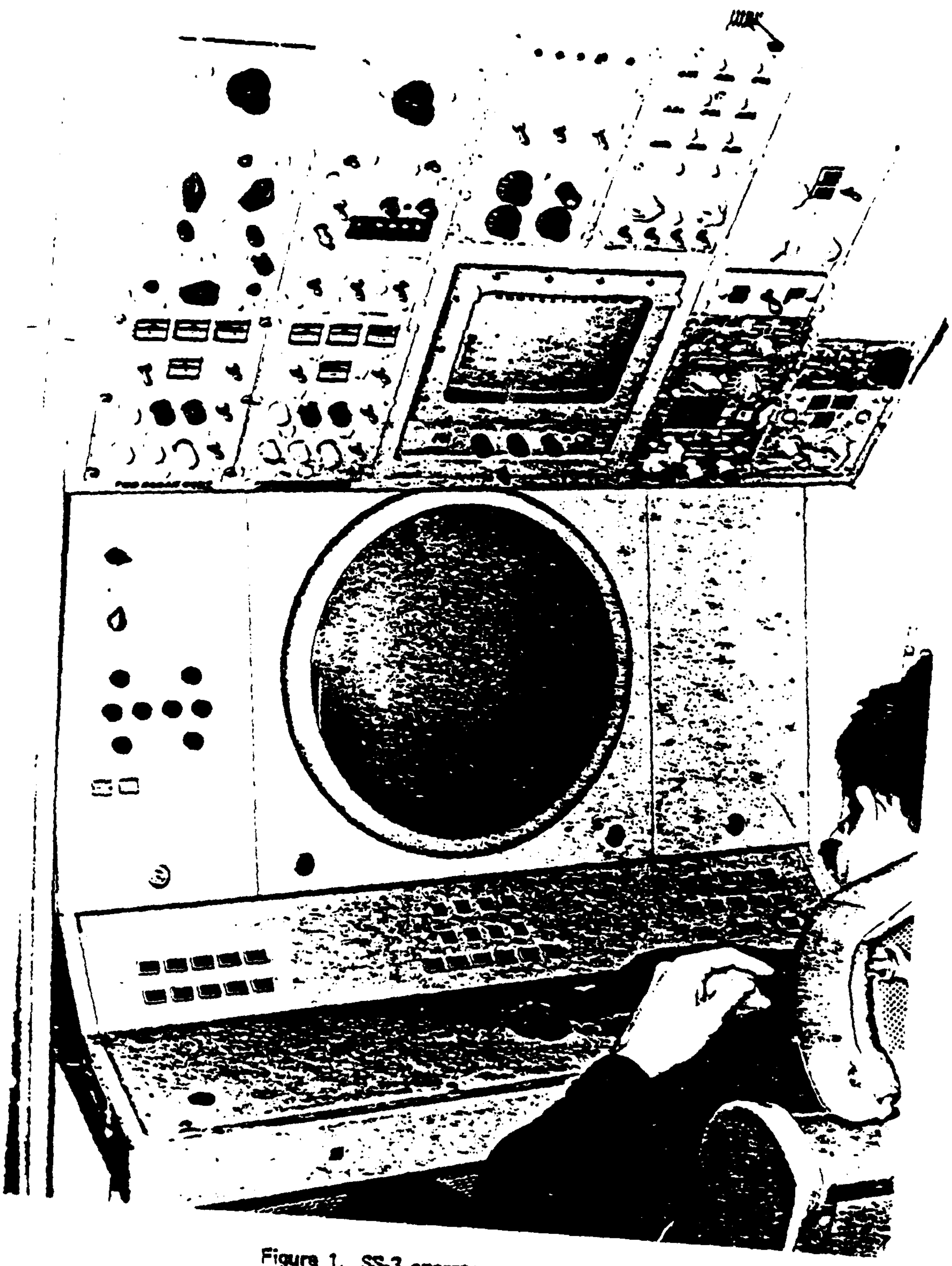


Figure 1. SS-3 operator station.

In the discussion that follows, the names that will be given to specific displays and controls in the operator's crewstation are shown in capital letters.

1.8.1 The SS-3 Operator Station

Figure 1 shows the SS-3 operator station on the P-3C. The primary display used by the operator is the multipurpose digital display (the RADAR-DISPLAY) on which both tactical symbology and raw radar data can be displayed. The controls that will be used are the TRACK-BALL and some momentary contact switches on the keyset tray -- the RADAR-MODE switch, the HOOK-VERIFY switch, and the ENTER-RADAR-CONTACT switch -- and a switch that is located on the Radar Control Panel, the LOAD switch.

1.8.2 Functions of the SS-3 Controls

If we assume that the radar equipment has been powered up and is functioning properly, then, when the operator depresses the RADAR-MODE switch, the system will light up a set of controls (the radar matrix) that indicates the subfunctions available to the operator. These controls permit the operator to perform a number of functions related to the processing radar data. We're only going to be concerned with one of these functions -- the ENTER-RADAR-CONTACT function (Figure 2). This control function enables the operator to enter the coordinates of a radar contact to the onboard computer. The computer will automatically assign a number to the contact, record the time at which the contact was entered, and display a permament symbol representing the contact on both the SS-3 operator's display and on the TACCO's display.

But in order to enter a radar contact, the operator must identify to the system which of the many potential contacts he is interested in. He does this by manipulating a TRACK-BALL which controls the position of a cursor on the screen. The cursor (the HOOK) appears on the screen as a small circle. The operator must manipulate the TRACK-BALL until the HOOK encircles the

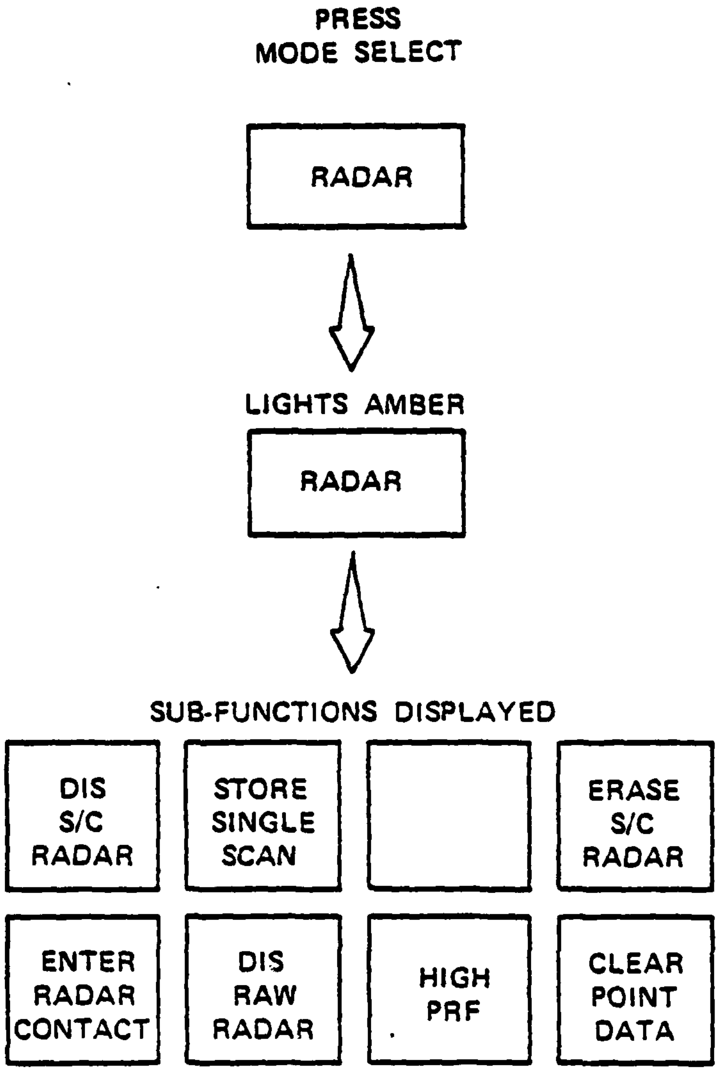


Figure 2. Radar matrix function.

MISSION

- PERFORM RADAR PLOT

  -- ENABLE THE RADAR-DISPLAY

  -- SEARCH FOR AN UNENTERED CONTACT

  -- IF ONE IS FOUND, ENTER IT BY:

     (1) MOVING THE HOOK TO THE RADAR CONTACT POSITION

     (2) DEPRESSING HOOK-VERIFY

     (3) ENABLE THE ENTER-RADAR-CONTACT FUNCTION, IF NECESSARY, BY DEPRESSING RADAR-MODE

     (4) DEPRESSING ENTER-RADAR-CONTACT

  -- IF NO MORE CAN BE FOUND, END

the particular symbol on the screen that he is interested in. Then, when he depresses the HOOK-VERIFY pushbutton, the system will cause the encircled symbol to blink. Any action that the operator then takes, such as depressing the ENTER-RADAR-CONTACT pushbutton, will be understood by the system to refer to the hooked symbol.

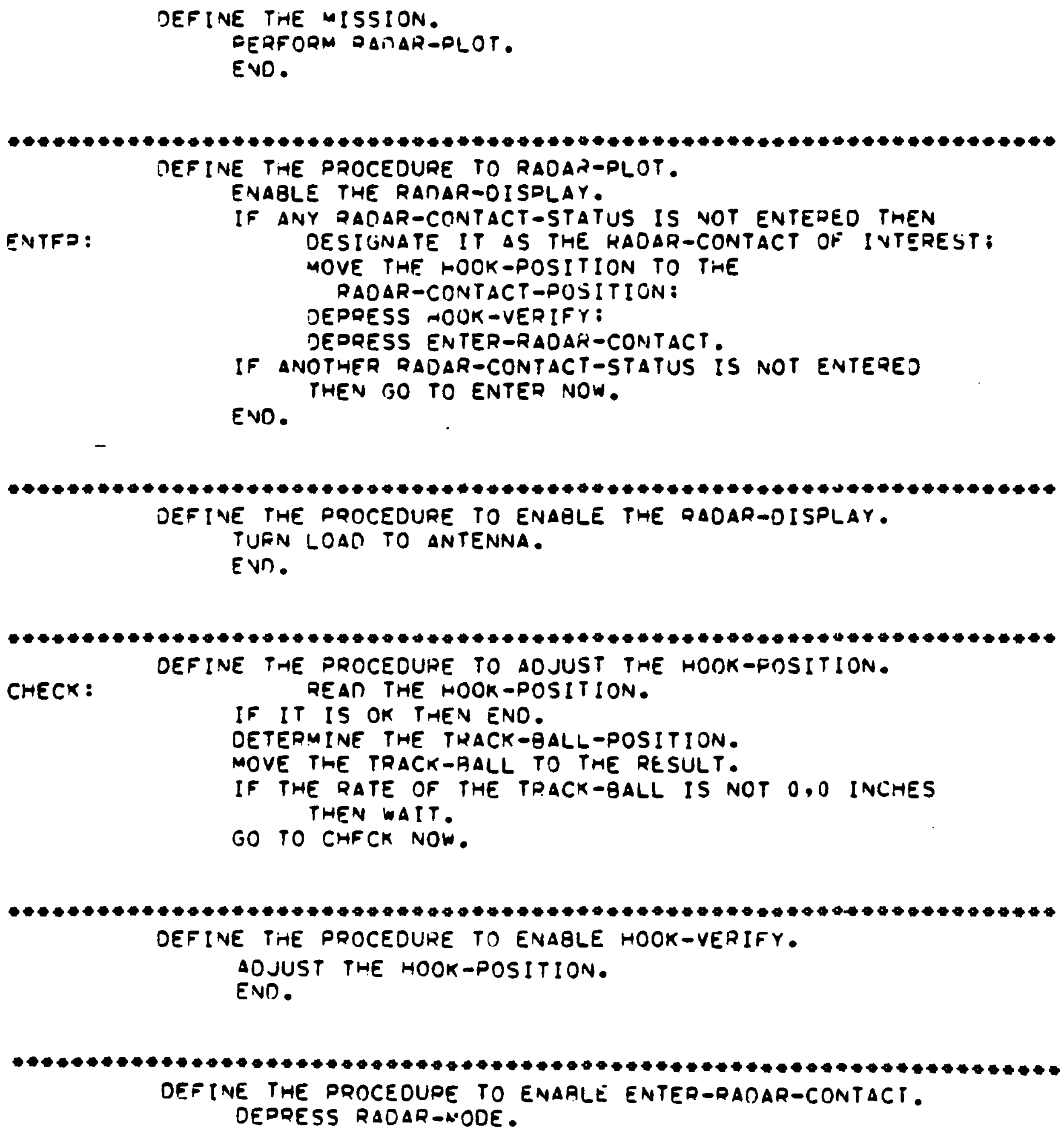
One issue that we have not discussed is how the operator powers up the radar equipment in the first place. For our purposes, we will assume that all that is required is for him to switch the LOAD switch from its dummy load to its antenna position -- all other radar initialization actions will be assumed to have been performed prior to the start of the simulation.

1.8.3 Outline of Operator Procedures

We can now outline what the operator is required to do in order to plot a set of radar targets. The operator's *mission*, in this case, is to perform a radar plot. In order to do this, the operator must first enable the radar matrix by depressing the radar mode pushbutton. Then he must search for an unentered contact. If he finds one, he must enter it by hooking the contact and depressing ENTER-RADAR-CONTACT. If he cannot find any more unentered contacts, he is done.

The HOS code that describes this mission is shown in Figure 3 . As you can see, there is a close correspondence between our verbal (outline) description of the operator's procedures and the code that HOS needs in order to simulate those procedures.

Examining the code, we see that HOS requires a definition of the mission. In this case, the mission is simply to perform a radar plot, although we could easily have added additional tasks to the mission. In a normal task analysis, we might simply dig through some references at this point to find an amount of time that we would use as the average amount of time required to perform a radar plot. Alternatively, we might pull a



```
               DEFINE THE MISSION.
                    PERFORM RADAR-PLOT.
                    END.

**************************************************************
               DEFINE THE PROCEDURE TO RADAR-PLOT.
                    ENABLE THE RADAR-DISPLAY.
                    IF ANY RADAR-CONTACT-STATUS IS NOT ENTERED THEN
ENTER:                   DESIGNATE IT AS THE RADAR-CONTACT OF INTEREST;
                         MOVE THE HOOK-POSITION TO THE
                           RADAR-CONTACT-POSITION;
                         DEPRESS HOOK-VERIFY;
                         DEPRESS ENTER-RADAR-CONTACT.
                    IF ANOTHER RADAR-CONTACT-STATUS IS NOT ENTERED
                         THEN GO TO ENTER NOW.
                    END.

**************************************************************
               DEFINE THE PROCEDURE TO ENABLE THE RADAR-DISPLAY.
                    TURN LOAD TO ANTENNA.
                    END.

**************************************************************
               DEFINE THE PROCEDURE TO ADJUST THE HOOK-POSITION.
CHECK:                   READ THE HOOK-POSITION.
                    IF IT IS OK THEN END.
                    DETERMINE THE TRACK-BALL-POSITION.
                    MOVE THE TRACK-BALL TO THE RESULT.
                    IF THE RATE OF THE TRACK-BALL IS NOT 0.0 INCHES
                         THEN WAIT.
                    GO TO CHECK NOW.

**************************************************************
               DEFINE THE PROCEDURE TO ENABLE HOOK-VERIFY.
                    ADJUST THE HOOK-POSITION.
                    END.

**************************************************************
               DEFINE THE PROCEDURE TO ENABLE ENTER-RADAR-CONTACT.
                    DEPRESS RADAR-MODE.
```

Figure 3. Operator procedures for the radar plotting simulation.

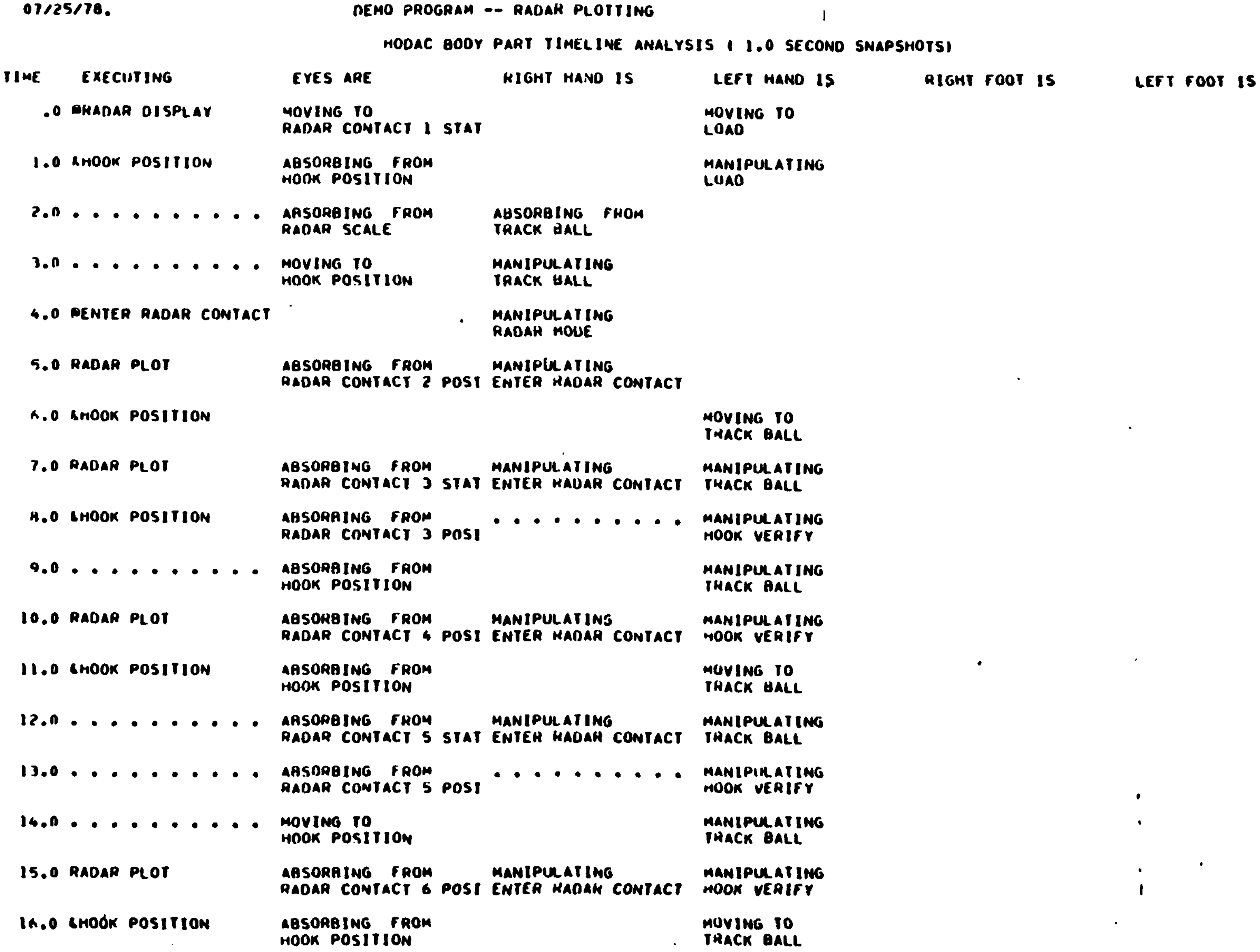
number to use out of the air. HOS would also permit you to do this, but it can instead estimate for you the amount of time that it *actually takes* to do the radar plotting, based on the number of contacts on the screen and the characteristics of the controls, both of which could vary considerably during a mission or from crewstation to crewstation. To do this, HOS requires an answer to the question: How does one do a radar plot?

The answer is in our outline of the operator's procedures -- the operator must first enable the radar matrix. How is this done? By depressing RADAR-MODE. He must then search for any radar-contact-symbol that is not entered and, if he finds one, enter it by moving the hook to the radar-contact's position, depressing HOOK-VERIFY, and depressing ENTER-RADAR-CONTACT. He must then serach for another radar contact that has not been entered and enter it in the same way. If there are no more to be entered, he's done.

1.8.4 The HOS Radar Plotting Simulation

Once these operator procedures have been combined with additional procedures that describe how the equipment functions, with information on the actual crewstation layout, and with specific target data, HOS can simulate the operator's functions within the crewstation. The data that HOS generates on the operator's performance can then be run through an analysis program, HODAC, that can produce eight different types of analyses and an almost infinite number of variants on these analyses according to the interests of the analyst. Examples of these analyses are shown in Figures 4 through 12 , to indicate the wealth of information available from HOS.

Figure 4 is an example of a Timeline Analysis. It documents what procedure the operator is working on at any instant and what each body part is doing. The granularity of the Timeline Analysis can be chosen by the analyst -- in this case, one second *snapshots* have been used. A related analysis is the Channel Loading Report (Figure 5) which indicates the



```
07/25/78.                        DEMO PROGRAM -- RADAR PLOTTING

                                    HODAC BODY PART TIMELINE ANALYSIS ( 1.0 SECOND SNAPSHOTS)

TIME  EXECUTING            EYES ARE             RIGHT HAND IS        LEFT HAND IS         RIGHT FOOT IS        LEFT FOOT IS

  .0 @RADAR DISPLAY       MOVING TO                                 MOVING TO
                          RADAR CONTACT 1 STAT                      LOAD

 1.0 &HOOK POSITION       ABSORBING  FROM                           MANIPULATING
                          HOOK POSITION                             LOAD

 2.0 . . . . . . . . . .  ABSORBING  FROM      ABSORBING  FROM
                          RADAR SCALE          TRACK BALL

 3.0 . . . . . . . . . .  MOVING TO            MANIPULATING
                          HOOK POSITION        TRACK BALL

 4.0 @ENTER RADAR CONTACT                      MANIPULATING
                                               RADAR MODE

 5.0 RADAR PLOT           ABSORBING  FROM      MANIPULATING
                          RADAR CONTACT 2 POSI ENTER RADAR CONTACT

 6.0 &HOOK POSITION                                                 MOVING TO
                                                                    TRACK BALL

 7.0 RADAR PLOT           ABSORBING  FROM      MANIPULATING         MANIPULATING
                          RADAR CONTACT 3 STAT ENTER RADAR CONTACT  TRACK BALL

 8.0 &HOOK POSITION       ABSORBING  FROM      . . . . . . . . . .  MANIPULATING
                          RADAR CONTACT 3 POSI                      HOOK VERIFY

 9.0 . . . . . . . . . .  ABSORBING  FROM                           MANIPULATING
                          HOOK POSITION                             TRACK BALL

10.0 RADAR PLOT           ABSORBING  FROM      MANIPULATING         MANIPULATING
                          RADAR CONTACT 4 POSI ENTER RADAR CONTACT  HOOK VERIFY

11.0 &HOOK POSITION       ABSORBING  FROM                           MOVING TO
                          HOOK POSITION                             TRACK BALL

12.0 . . . . . . . . . .  ABSORBING  FROM      MANIPULATING         MANIPULATING
                          RADAR CONTACT 5 STAT ENTER RADAR CONTACT  TRACK BALL

13.0 . . . . . . . . . .  ABSORBING  FROM      . . . . . . . . . .  MANIPULATING
                          RADAR CONTACT 5 POSI                      HOOK VERIFY

14.0 . . . . . . . . . .  MOVING TO                                 MANIPULATING
                          HOOK POSITION                             TRACK BALL

15.0 RADAR PLOT           ABSORBING  FROM      MANIPULATING         MANIPULATING
                          RADAR CONTACT 6 POSI ENTER RADAR CONTACT  HOOK VERIFY

16.0 &HOOK POSITION       ABSORBING  FROM                           MOVING TO
                          HOOK POSITION                             TRACK BALL
```

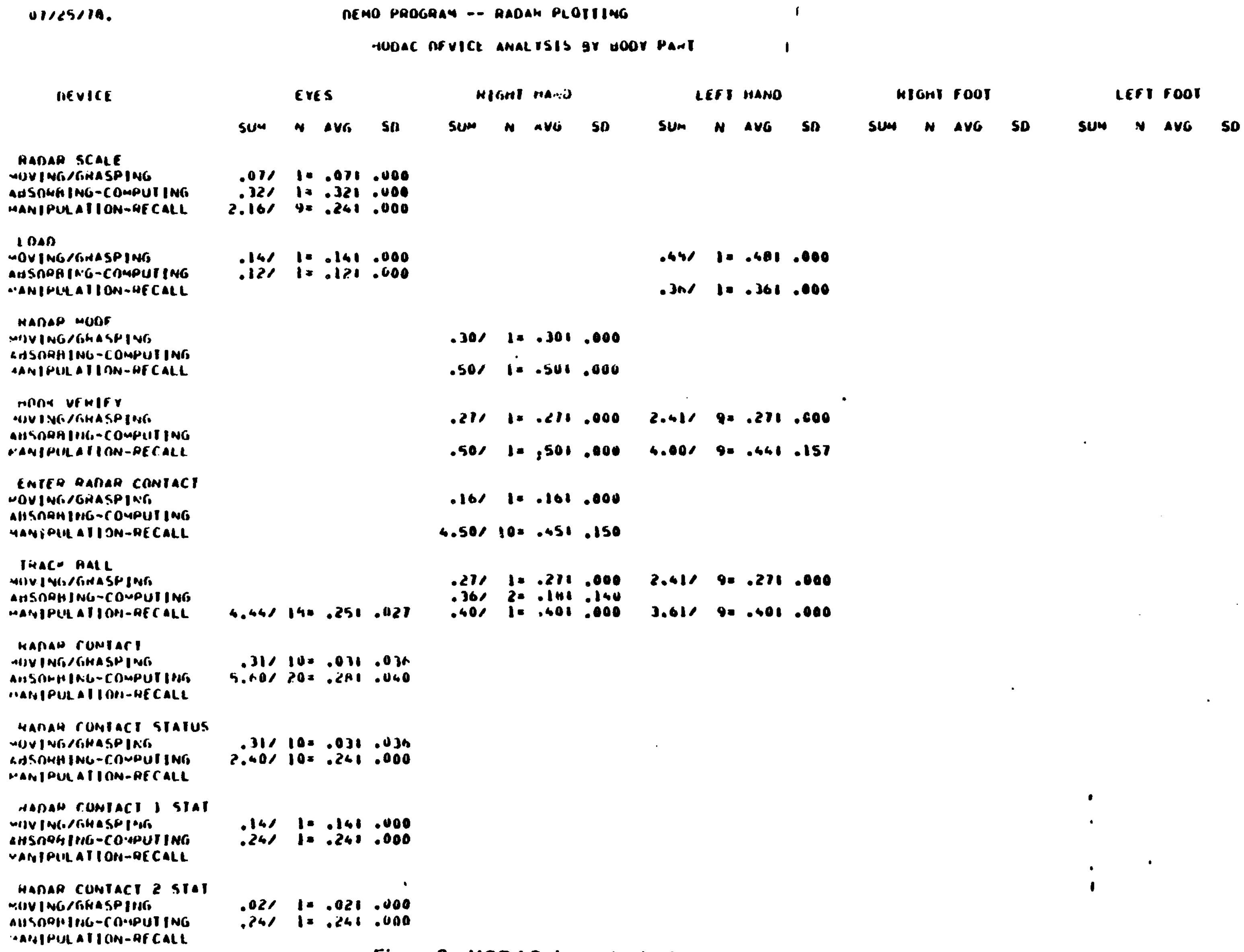
Figure 4. HODAC timeline analysis.

```
09/11/78.                    DEMO PROGRAM -- RADAR PLOTTING                                PAGE   1

                                  HODAC CHANNEL LOADING REPORT ( 1.0 SNAPSHOTS)

 TIME       EYES               MENTAL           RIGHT HAND          LEFT HAND        RIGHT FOOT         LEFT FOOT

  1.00 51.2  *****       21.4  **           .0                76.1  ********      .0                .0


  2.00 81.1  ********    72.6  *******    17.9  **             5.7  *             .0                .0
  3.00 14.6  *           40.0  ****       79.4  ********        .0                .0                .0
  4.00   .0                .0             99.0  **********      .0                .0                .0
  5.00 25.5  ***         23.6  **         99.0  **********      .0                .0                .0
  6.00 22.3  **          92.4  *********  24.2  **              .0                .0                .0
  7.00 11.3  *           21.4  **         74.7  *******         .0                .0                .0
  8.00 32.3  ***         48.0  *****      62.7  ******          .0                .0                .0
  9.00 31.3  ***         27.3  ***        50.7  *****           .0                .0                .0
 10.00 52.4  *****       60.6  ******     82.7  ********      50.0  *****         .0                .0
 11.00   .0              60.1  ******       .0                35.9  ****          .0                .0
 12.00 36.4  ****        45.4  *****        .0                45.6  *****         .0                .0
 13.00 14.6  *           46.6  *****        .0                55.4  ******        .0                .0
 14.00 32.3  ***         24.6  ***        23.4  **            88.2  *********     .0                .0
 15.00 47.8  *****       74.6  *******    26.6  ***           26.6  ***           .0                .0
 16.00   .8              12.0  *            .0                63.2  ******        .0                .0
 17.00 53.2  *****       44.0  ****       39.6  ****          83.3  ********      .0                .0
 18.00  9.8  *           94.0  *********  10.4  *             36.4  ****          .0                .0
 19.00 13.3  *           25.5  ***         3.5                88.2  *********     .0                .0
 20.00 75.2  ********    77.3  ********   46.5  *****         46.5  *****         .0                .0
 21.00   .8              29.2  ***          .0                70.8  *******       .0                .0
 22.00 53.2  *****       44.0  ****       39.7  ****          83.5  ********      .0                .0
 23.00  9.5  *           80.0  ********   10.3  *             23.8  **            .0                .0
 24.00 10.1  *           36.3  ****         .0                63.9  ******        .0                .0
 25.00 23.9  **          30.5  ***          .0                58.1  ******        .0                .0
 26.00 32.8  ***         37.2  ****         .0                62.0  ******        .0                .0
 27.00 47.7  *****       75.2  ********   50.0  *****         69.1  *******       .0                .0
 28.00   .8              28.8  ***          .0                55.2  ******        .0                .0
 29.00 45.8  *****       41.4  ****       36.2  ****          88.2  *********     .0                .0
 30.00 39.1  ****        74.5  *******    13.8  *             21.7  **            .0                .0
 31.00 14.6  *           24.0  **           .0                83.3  ********      .0                .0
 32.00 47.6  *****       74.5  *******    50.0  *****         69.9  *******       .0                .0
 33.00   .8              41.5  ****         .0                54.5  *****         .0                .0
 34.00 36.5  ****        48.0  *****        .0                45.6  *****         .0                .0
 35.00 14.6  *           44.8  ****         .0                55.4  ******        .0                .0
```

Figure 5. HODAC channel loading report.

percentage of time within each snapshot interval that each body part is occupied. The Devices by Body Part Analysis (Figure 6) tabulates the total amount of time each device was used by each body part and calculates the means and standard deviations for each of three basic functions -- moving the device, reading information from the device, and performing a control manipulation on the device. The Devices by Usage Analysis (Figure 7) provides summary data on some of the other types of functions the operator may be performing on each device. The Devices by Procedure Analysis (Figure 8) provides usage data statistics by procedure. The Procedural Analysis (Figure 9) summarizes this data over procedures. The Label Analysis (Figure 10) provides summary data for procedures and certain types of within-procedure statistics. And, finally, the Link Analysis (Figures 11 through 12) provides data on the frequencies of usage of groups of displays and controls and transitions from one group to another.

Section 4 will describe in more detail how each of these reports is to be interpreted and used.

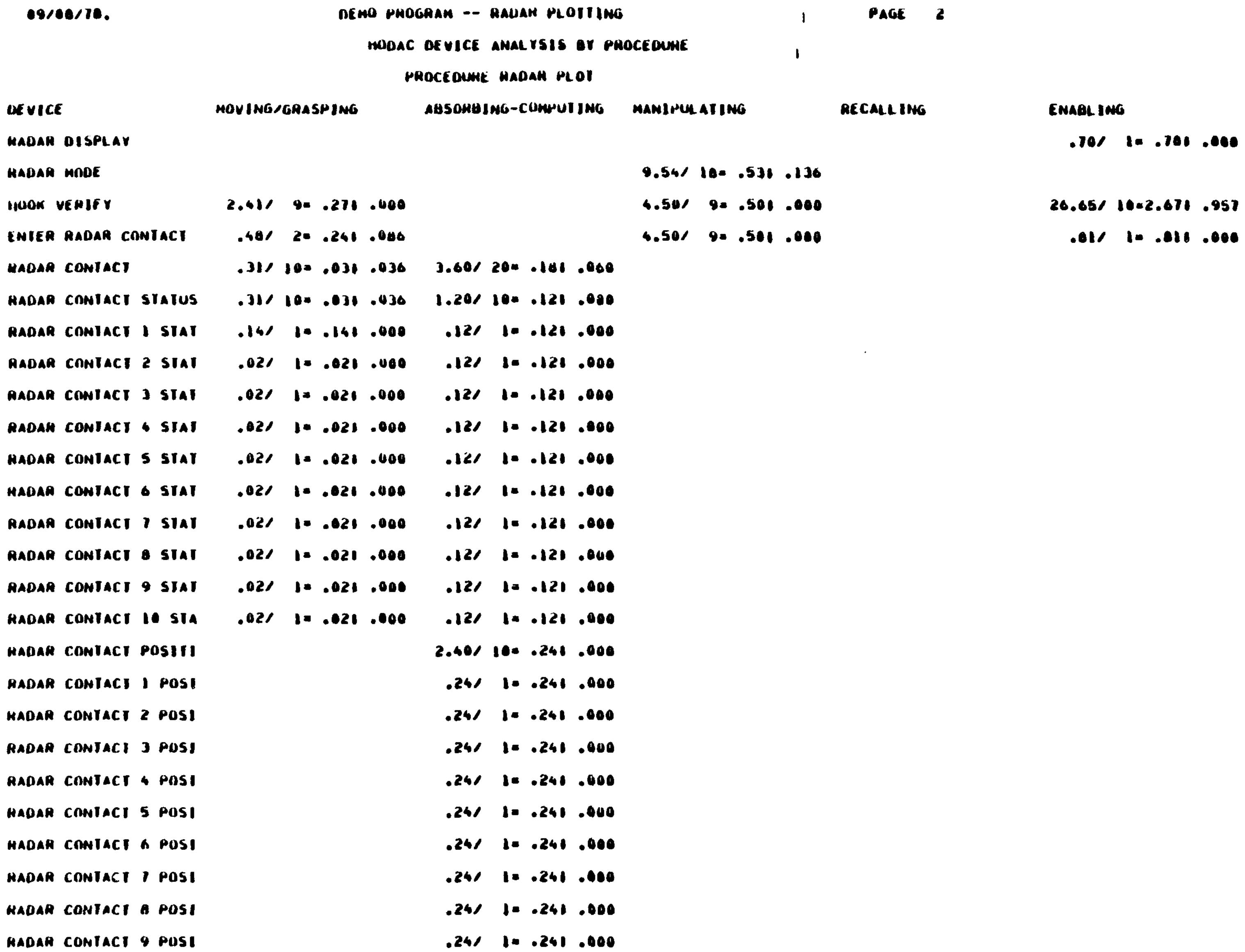


```
07/25/78.                    DEMO PROGRAM -- RADAR PLOTTING

                               HODAC DEVICE ANALYSIS BY BODY PART

   DEVICE                  EYES                RIGHT HAND              LEFT HAND              RIGHT FOOT             LEFT FOOT

                     SUM   N   AVG   SD     SUM   N   AVG   SD     SUM   N   AVG   SD     SUM   N   AVG   SD     SUM   N   AVG   SD

  RADAR SCALE
 MOVING/GRASPING       .07/  1= .071 .000
 ABSORBING-COMPUTING   .32/  1= .321 .000
 MANIPULATION-RECALL  2.16/  9= .241 .000

  LOAD
 MOVING/GRASPING       .14/  1= .141 .000                          .48/  1= .481 .000
 ABSORBING-COMPUTING   .12/  1= .121 .000
 MANIPULATION-RECALL                                               .36/  1= .361 .000

  RADAR MODE
 MOVING/GRASPING                             .30/  1= .301 .000
 ABSORBING-COMPUTING
 MANIPULATION-RECALL                         .50/  1= .501 .000

  HOOK VERIFY
 MOVING/GRASPING                             .27/  1= .271 .000   2.41/  9= .271 .000
 ABSORBING-COMPUTING
 MANIPULATION-RECALL                         .50/  1= ;501 .000   4.00/  9= .441 .157

  ENTER RADAR CONTACT
 MOVING/GRASPING                             .16/  1= .161 .000
 ABSORBING-COMPUTING
 MANIPULATION-RECALL                        4.50/ 10= .451 .150

  TRACK BALL
 MOVING/GRASPING                             .27/  1= .271 .000   2.41/  9= .271 .000
 ABSORBING-COMPUTING                         .36/  2= .181 .140
 MANIPULATION-RECALL  4.44/ 14= .251 .027    .40/  1= .401 .000   3.61/  9= .401 .000

  RADAR CONTACT
 MOVING/GRASPING       .31/ 10= .031 .036
 ABSORBING-COMPUTING  5.60/ 20= .281 .040
 MANIPULATION-RECALL

  RADAR CONTACT STATUS
 MOVING/GRASPING       .31/ 10= .031 .036
 ABSORBING-COMPUTING  2.40/ 10= .241 .000
 MANIPULATION-RECALL

  RADAR CONTACT 1 STAT
 MOVING/GRASPING       .14/  1= .141 .000
 ABSORBING-COMPUTING   .24/  1= .241 .000
 MANIPULATION-RECALL

  RADAR CONTACT 2 STAT
 MOVING/GRASPING       .02/  1= .021 .000
 ABSORBING-COMPUTING   .24/  1= .241 .000
 MANIPULATION-RECALL
```

Figure 6. HODAC de[illegible] by body part analysis.

```
07/25/78.                    DEMO PROGRAM -- RADAR PLOTTING

                                HODAC DEVICE ANALYSIS BY USAGE

DEVICE                 MOVING/GRASPING       ABSORBING-COMPUTING   MANIPULATING         RECALLING            ENABLING

RADAR DISPLAY                                                                                                   .74/  1= .74! .000

RADAR SCALE             .07/  1= .07! .000    .32/  1= .32! .000                        2.16/  9= .24! .000

LOAD                    .62/  2= .31! .170    .12/  1= .12! .000    .36/  1= .36! .000

RADAR MODE              .30/  1= .30! .000                          .50/  1= .50! .000

HOOK VERIFY            2.68/ 10= .27! .000                         4.50/  9= .50! .000                         .00/ 20= .00! .000

ENTER RADAR CONTACT     .16/  1= .16! .000                         4.50/  9= .50! .000                         .81/  1= .81! .000

TRACK BALL             2.68/ 10= .27! .000    .36/  2= .18! .140   4.02/ 10= .40! .000  4.44/ 18= .25! .027

RADAR CONTACT           .31/ 10= .03! .036   5.60/ 20= .28! .040

RADAR CONTACT STATUS    .31/ 10= .03! .036   2.40/ 10= .24! .000

RADAR CONTACT 1 STAT    .14/  1= .14! .000    .24/  1= .24! .000

RADAR CONTACT 2 STAT    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT 3 STAT    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT 4 STAT    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT 5 STAT    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT 6 STAT    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT 7 STAT    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT 8 STAT    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT 9 STAT    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT 10 STA    .02/  1= .02! .000    .24/  1= .24! .000

RADAR CONTACT POSITI                         3.20/ 10= .32! .000

RADAR CONTACT 1 POSI                          .32/  1= .32! .000

RADAR CONTACT 2 POSI                          .32/  1= .32! .000

RADAR CONTACT 3 POSI                          .32/  1= .32! .000

RADAR CONTACT 4 POSI                          .32/  1= .32! .000

RADAR CONTACT 5 POSI                          .32/  1= .32! .000

RADAR CONTACT 6 POSI                          .32/  1= .32! .000
```

Figure 7. HODAC devices by usage analysis.

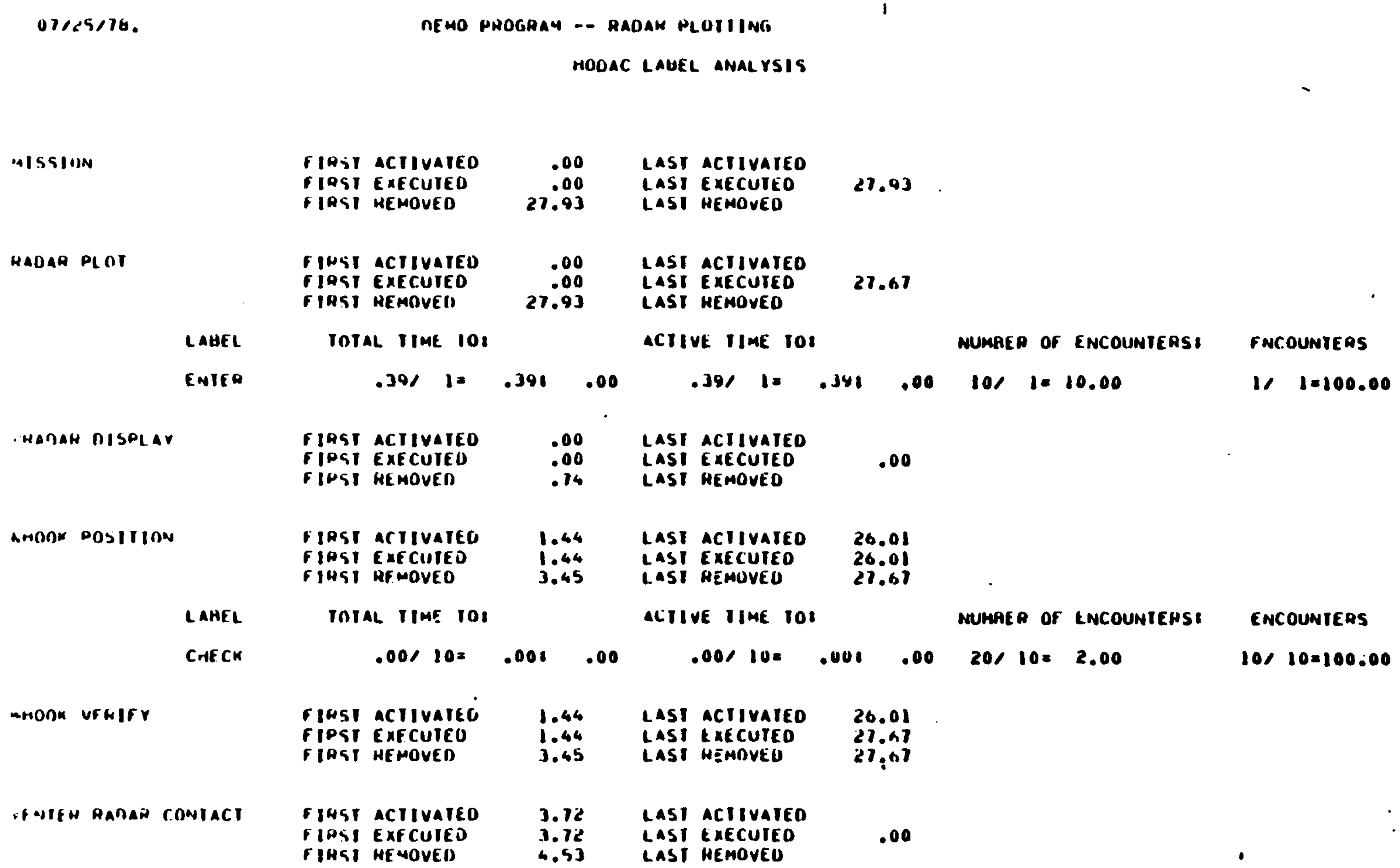


```
09/08/78.                    DEMO PROGRAM -- RADAR PLOTTING              PAGE   2
                              HODAC DEVICE ANALYSIS BY PROCEDURE
                                  PROCEDURE RADAR PLOT
DEVICE                  MOVING/GRASPING      ABSORBING-COMPUTING  MANIPULATING        RECALLING           ENABLING
RADAR DISPLAY                                                                                              .70/  1= .70% .000
RADAR MODE                                                        9.54/ 10= .53% .136
HOOK VERIFY              2.41/  9= .27% .000                      4.50/  9= .50% .000                     26.65/ 10=2.67% .957
ENTER RADAR CONTACT       .48/  2= .24% .086                      4.50/  9= .50% .000                       .81/  1= .81% .000
RADAR CONTACT             .31/ 10= .03% .036  3.60/ 20= .18% .060
RADAR CONTACT STATUS      .31/ 10= .03% .036  1.20/ 10= .12% .000
RADAR CONTACT 1 STAT      .14/  1= .14% .000   .12/  1= .12% .000
RADAR CONTACT 2 STAT      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT 3 STAT      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT 4 STAT      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT 5 STAT      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT 6 STAT      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT 7 STAT      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT 8 STAT      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT 9 STAT      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT 10 STA      .02/  1= .02% .000   .12/  1= .12% .000
RADAR CONTACT POSITI                          2.40/ 10= .24% .000
RADAR CONTACT 1 POSI                           .24/  1= .24% .000
RADAR CONTACT 2 POSI                           .24/  1= .24% .000
RADAR CONTACT 3 POSI                           .24/  1= .24% .000
RADAR CONTACT 4 POSI                           .24/  1= .24% .000
RADAR CONTACT 5 POSI                           .24/  1= .24% .000
RADAR CONTACT 6 POSI                           .24/  1= .24% .000
RADAR CONTACT 7 POSI                           .24/  1= .24% .000
RADAR CONTACT 8 POSI                           .24/  1= .24% .000
RADAR CONTACT 9 POSI                           .24/  1= .24% .000
```

Figure 8. HODAC dev[illegible] by procedure analysis.

```
07/25/78.                   DEMO PROGRAM -- RADAR PLOTTING

                                HODAC PROCEDURAL ANALYSIS

PROCEDURE            MOVING/GRASPING      ABSORBING            MANIPULATING         RECALLING            ENABLING

RADAR PLOT            3.14/ 21= .15! .118   5.60/ 20= .28! .040   9.00/ 18= .50! .000                        1.54/ 22= .07! .222
*RADAR DISPLAY         .62/  2= .31! .170    .12/  1= .12! .000    .36/  1= .36! .000
*HOOK POSITION        3.17/ 31= .10! .115   2.56/ 43= .06! .071   4.02/ 10= .40! .000   6.60/ 27= .24! .023
*ENTER RADAR CONTACT   .30/  1= .30! .000                          .50/  1= .50! .000
```

Figure 9. HODAC procedural analysis.



```
07/25/78.                          DEMO PROGRAM -- RADAR PLOTTING

                                              HODAC LABEL ANALYSIS


MISSION                   FIRST ACTIVATED       .00    LAST ACTIVATED
                          FIRST EXECUTED        .00    LAST EXECUTED     27.93
                          FIRST REMOVED       27.93    LAST REMOVED

RADAR PLOT                FIRST ACTIVATED       .00    LAST ACTIVATED
                          FIRST EXECUTED        .00    LAST EXECUTED     27.67
                          FIRST REMOVED       27.93    LAST REMOVED

               LABEL        TOTAL TIME IO:             ACTIVE TIME TO:             NUMBER OF ENCOUNTERS:    ENCOUNTERS

               ENTER           .39/  1=   .39I   .00      .39/  1=   .39I   .00   10/  1= 10.00            1/  1=100.00

*RADAR DISPLAY            FIRST ACTIVATED       .00    LAST ACTIVATED
                          FIRST EXECUTED        .00    LAST EXECUTED       .00
                          FIRST REMOVED         .74    LAST REMOVED

*HOOK POSITION            FIRST ACTIVATED      1.44    LAST ACTIVATED    26.01
                          FIRST EXECUTED       1.44    LAST EXECUTED     26.01
                          FIRST REMOVED        3.45    LAST REMOVED      27.67

               LABEL        TOTAL TIME TO:             ACTIVE TIME TO:             NUMBER OF ENCOUNTERS:    ENCOUNTERS

               CHECK           .00/ 10=   .00I   .00      .00/ 10=   .00I   .00   20/ 10=  2.00           10/ 10=100.00

*HOOK VERIFY              FIRST ACTIVATED      1.44    LAST ACTIVATED    26.01
                          FIRST EXECUTED       1.44    LAST EXECUTED     27.67
                          FIRST REMOVED        3.45    LAST REMOVED      27.67

*ENTER RADAR CONTACT      FIRST ACTIVATED      3.72    LAST ACTIVATED
                          FIRST EXECUTED       3.72    LAST EXECUTED       .00
                          FIRST REMOVED        4.53    LAST REMOVED
```

Figure 10. HODAC label analysis.

```
09/09/78.                 DEMO PROGRAM -- RADAR PLOTTING                         PAGE   2

                                           HODAC LINK ANALYSIS
                                             LINK MOVEMENTS

                                                        TO CONTROLS

       FROM                   EYES              RIGHT HAND           LEFT HAND            RIGHT FOOT           LEFT FOOT
 ________________    ________________    ________________    ________________    ________________    ________________
CRT
   MOVEMENT TIME
   TOTAL IDLE/DWELL
   ACTIVE TIME
   IDLE TIME
CONTROLS
   MOVEMENT TIME                              .33/  1= .33! .000   1.87/  7= .27! .000
   TOTAL IDLE/DWELL                          3.90/  1=3.90! .000  19.94/  7=2.85! .976
   ACTIVE TIME                                .00/  1= .00! .000    .00/  7= .00! .000
   IDLE TIME                                 3.90/  1=3.90! .000  19.94/  7=2.85! .976
RELAXATION
   MOVEMENT TIME         .14/  1= .14! .000   .27/  1= .27! .000    .75/  2= .37! .105
   TOTAL IDLE/DWELL      .00/  1= .00! .000  1.A2/  1=1.82! .000   4.33/  2=2.16!1.944
   ACTIVE TIME           .00/  1= .00! .000   .00/  1= .00! .000    .00/  2= .00! .000
   IDLE TIME             .00/  1= .00! .000  1.A2/  1=1.82! .000   4.33/  2=2.16!1.944
```

Figure 11. HODAC link analysis.

```
09/09/78.                DEMO PROGRAM -- RADAR PLOTTING                       PAGE   4

                                        HODAC LINK ANALYSIS
                                         LINK FREQUENCIES

                         EYES(    39)     R. HAND(     2)    L. HAND(    10)    R. FOOT(     0)    L. FOOT(     0)
                         ------------     ------------      ------------      ------------      ------------
CRT
     ALL   CRT            37 / 94.872%

     TO    CONTROLS
     FROM CONTROLS         1 /  2.564%
     ALL   CONTROLS        1 /  2.564%

     TO    RELAXATION
     FROM RELAXATION
     ALL   RELAXATION

CONTROLS
     ALL   CONTROLS                          1 / 50.000%        7 / 70.000%

     TO    CRT             1 /  2.564%
     FROM CRT
     ALL   CRT             1 /  2.564%

     TO    RELAXATION                                           1 / 10.000%
     FROM RELAXATION       1 /  2.564%       1 / 50.000%        2 / 20.000%
     ALL   RELAXATION      1 /  2.564%       1 / 50.000%        3 / 30.000%

RELAXATION
     ALL   RELAXATION

     TO    CRT
     FROM CRT
     ALL   CRT

     TO    CONTROLS        1 /  2.564%       1 / 50.000%        2 / 20.000%
     FROM CONTROLS                                              1 / 10.000%
     ALL   CONTROLS        1 /  2.564%       1 / 50.000%        3 / 30.000%
```

Figure 12. HODAC link analysis.

This page is intentionally
left blank

INFORMATION ABSORPTION

- MODALITIES ARE EYES, HANDS, AND FEET
- HEARING, SPEECH, AND KINESTHETIC CUES ARE NOT MODELED
- MODALITIES ARE ASSOCIATED WITH DEVICES
- OCCURS AS A SERIES OF MICRO-ABSORPTIONS THAT INCREASE CONFIDENCE (HAB STRENGTH)

2. THE HOS OPERATOR MODELS

2.1 INFORMATION ABSORPTION

2.1.1 Absorption Modalities

The HOS operator has three modalities by which he can obtain information -- his eyes, hands, and feet. Currently, neither hearing nor speech nor any kinesthetic cues, such as vibration, balance, or the perception of external motions are simulated. This is not to say that these types of cues don't exist -- rather, we did not feel that there are currently any satisfactory models for the effects that these factors have on an operator's performance that operated on a level such that we could include them in HOS.

When describing the displays and controls in the operator's crewstation, the analyst must identify the modality (eyes, hands, or feet) that the operator is to use when obtaining information from each device. Thus, if the analyst were describing the displays and controls in an automobile, he would indicate to HOS that the operator is to use his eyes to read the fuel gauge, his hands to "read" the steering wheel, and his foot to "read" the accelerator.

The process by which the operator obtains information is the same for each modality and consists of a series of *micro-absorptions*. Each micro-absorption requires time. As the operator spends more time (more micro-absorptions) reading a device, both his knowledge of the device's value and his confidence in that knowledge increase until his confidence exceeds a threshold, at which time the absorption process is terminated.*

*Several other conditions may cause an absorption to be terminated, as will be discussed below.

2.1.2 Absorption Hab Strengths

The quantity that represents the operator's confidence in his knowledge of the value of a device is termed *hab strength*, after the learning theory concept called "habit strength" by Clark Hull. Each device has an associated hab strength that builds up during absorption. As the operator spends more time absorbing information, the hab strength associated with that information increases until it exceeds a threshold value, at which point absorption is terminated.

As an example, assume that the operator is attempting to read a device (e.g., a warning light) that has two discrete settings -- on and off. Successive micro-absorptions will cause the hab strength to increase as shown in the top curve in Figure 13 -- so that in 3-4 micro-absorptions, the operator has, for all intents and purposes, established in his own mind whether the display is on or off. Using a basic micro-absorption time charge of .04, such a read operation would require .12-.16 seconds (as compared to an average rate for reading words of approximately .18 seconds per word). For a display that is more difficult to read, more micro-absorptions are required to reach a comparable hab strength, as in the second curve for which the micro-absorption time charge was .12. Similarly, displays that have more potential values -- i.e., displays with more settings or continuous displays -- require still more micro-absorptions in order to reach a comparable hab strength. The bottom two curves shown in Figure 13, for example, represent the increase in hab strength for two continuous displays with micro-absorption time charges equal to those of the two displays in the upper curves. It should be noted that the equations used for continuous displays are the same as those for discrete displays with seven or more settings -- i.e., discrete displays with more than seven settings are treated as if they were continuous.

Figure 14 shows the effect that the micro-absorption time charge can have on the amount of time spent in a single,complete (macro-) absorption. The four curves represent the same four displays as in the preceding figure.

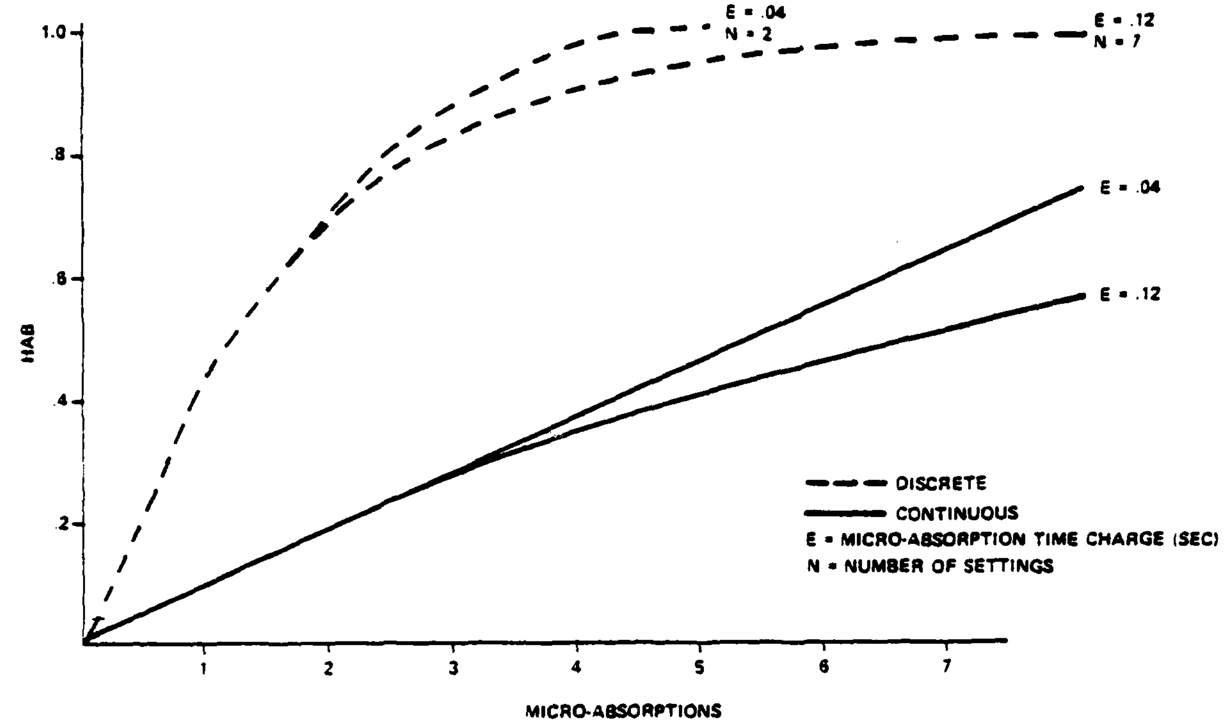


Figure 13. Hab strength as a function of the number of micro absorptions.

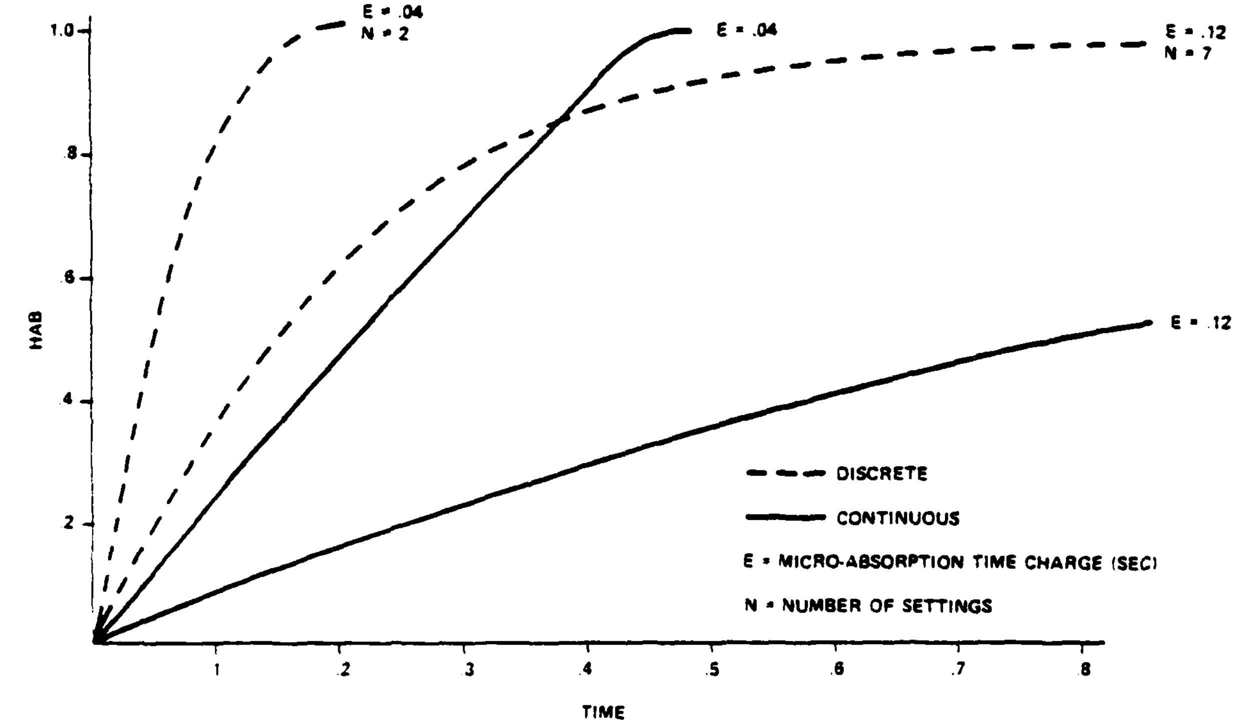


Figure 14. Hab strength as a function of absorption time.

In Figure 14, however, it can be seen that if the operator spends as much as .4 seconds in the absorption process, the hab strength for the "easy-to-read" continuous display will exceed the hab strength for the difficult discrete display.

The primary criterion for terminating an absorption process is for the hab strength of the device value being absorbed to exceed a threshold value, but there are several other conditions that the analyst can impose as input parameters that will terminate absorption. These conditions are:

- A maximum amount of time to be spent absorbing.
- A maximum number of micro-absorptions.
- A tolerance value that specifies that the hab strength has reached an asymptote.
- A tolerance on the accuracy to which the operator is required to read any device -- after which he is considered to "know" the value of the device.

The interaction between these termination conditions are discussed in more detail in the Appendix.

2.1.3 Absorption Estimates and Errors

During the absorption process, the operator acquires knowledge about the value of a device and confidence in his knowledge of that value. The value that the operator believes a continuous device to have (the *estimated* value of the device) is determined from the *actual* value of the device by adding an error term that is normally distributed about the actual value and whose magnitude is dependent upon an *accuracy* for the device as supplied by the analyst. Thus, if the analyst has specified that a particular device can be read to an accuracy of two percent, then two percent of the actual value of the device is used as the standard deviation when computing the value that the operator believes the device to have on any specific absorption.

ABSORPTION ESTIMATES

-- NORMALLY DISTRIBUTED ABOUT THE ACTUAL VALUE OF THE DEVICE

-- STANDARD DEVIATION IS SUPPLIED BY ANALYST AS A PERCENT

-- E.G., A DISPLAY CAN BE READ ± 2%

INFORMATION RECALL

LONG TERM RECALL OF

-- DEVICE LOCATIONS
-- DEVICE CHARACTERISTICS
-- PROCEDURES
-- MENTAL CALCULATION PROCESSES

SHORT TERM RECALL OF

-- CURRENT DEVICE VALUES

SHORT TERM MEMORY

- LINKED TO PERCEPTION VIA HAB STRENGTH
- PROBABILITY OF RECALL = $H^{\sqrt{t}}$
- REGION OF "NEAR-RECALL"
- EFFECT OF REHEARSAL

2.1.4 Accessing the Information Absorption Micro-Model

The analyst can force the HOS operator to read the value of a device by means of the statement:

READ device

In addition, HOS will automatically cause the operator to read the device whenever the operator needs its value and can't remember it, even if there is no explicit READ statement.

2.2 INFORMATION RECALL

2.2.1 Long-Term and Short-Term Memory

The HOS information recall model consists of two submodels -- a model for short-term memory and one for long-term memory. The long-term memory model is currently limited to the recall of certain types of predetermined information. Specifically, the HOS operator is assumed to have a completely *accurate* and *instantaneous* recall of the *locations* of all the displays and controls in his crewstation, most of their *characteristics*, the *procedures* that must be followed in carrying out a job, and the *calculation processes* for any mental computations that must be performed. These assumptions are consonant with the basic assumption of the HOS model -- namely, that the operator being simulated is a *trained* operator who performs routine operations automatically.

The short-term memory model is more elaborate. Short-term memory is considered to be linked to perception through the hab strength concept. As explained above, during the process of absorbing information, the operator's confidence in his knowledge of the value of a device increases until it exceeds a threshold at which point the absorption process is terminated. The ultimate hab strength associated with the device, a value between zero and one, constitutes a measure of the operator's confidence in his knowledge of the device value.

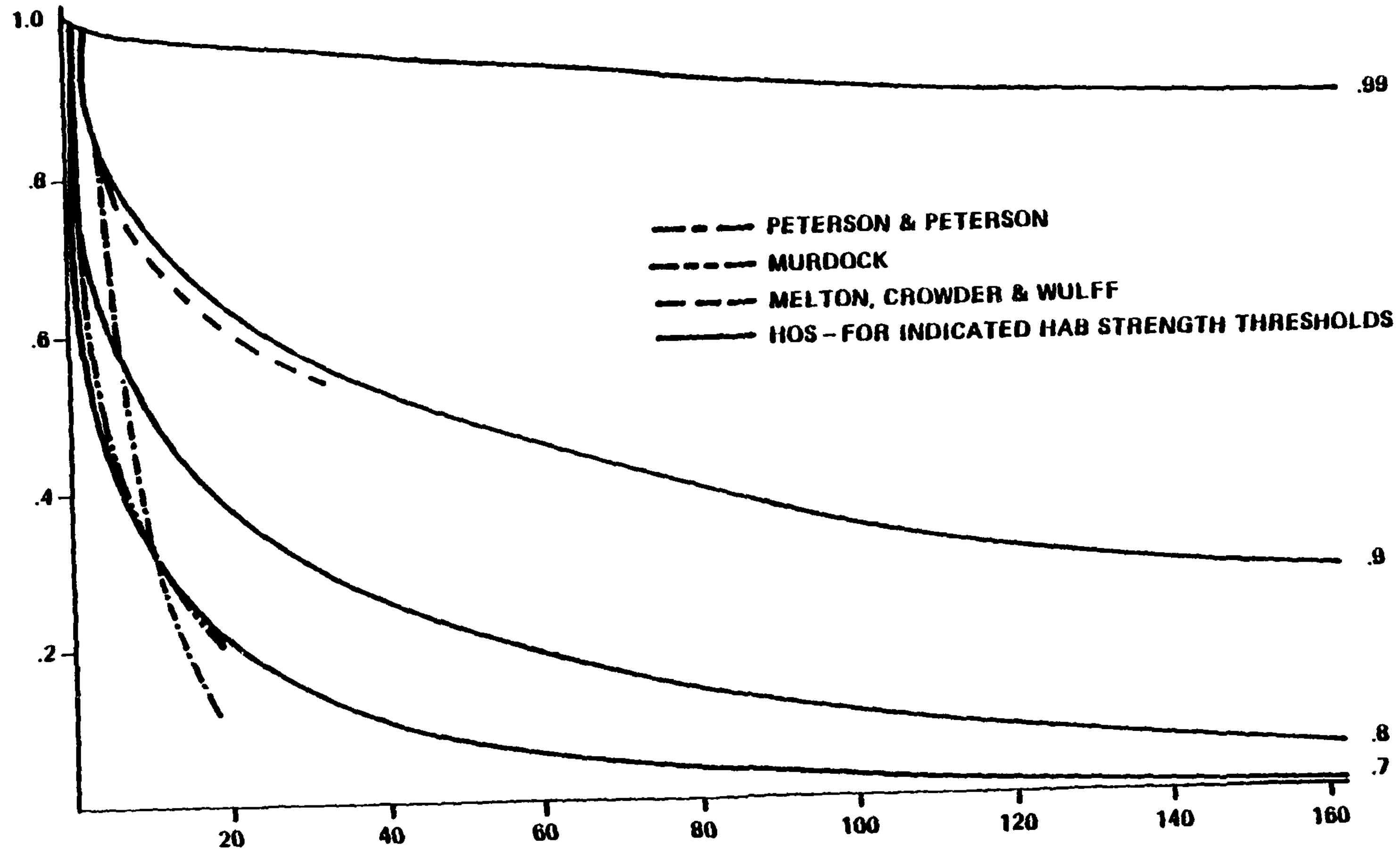


Figure 15. Experimental data on short-term memory compared with HOS Hab strength recall probabilities.

During recall, the hab strength is used to determine the probability that the operator will recall information absorbed from a device. The probability of successful recall is given by:

$$P = H^{\sqrt{t}}$$

where H is the hab strength and t is the time in seconds since the last absorption. Since H is a value between zero and one, the probability of recall is one at time zero -- i.e., the operator has an instantaneous memory of the value of a device that is perfect, to the extent that he learns the information in the first place. One second after the completion of an absorption, the probability of recall is exactly equal to the hab strength. As soon as absorption is complete, the probability of recall begins to decay exponentially as shown in Figure 15. Thus, within 60 seconds after an absorption that had raised the hab strength to .7, the probability of successful recall would be less than .1. If, however, the hab strength had been raised to .9, the probability of recall would stay above the level .1 for approximately seven minutes. Figure 15 shows recall probabilities from some of the available experimental data on short-term memory and how these data correspond to various hab strength values. Based on these data, we have chosen .8 as the default value for the hab strength threshold -- the value that is used to determine when the absorption process is terminated.

The value of P from the probability of recall equation:

$$P = H^{\sqrt{t}}$$

is compared against a number drawn at random from a uniform distribution. If the randomly drawn number is less than P, then the information is "remembered." If the randomly drawn number is much larger than P, then the information is "forgotten." If, however, the randomly drawn number is close to P, then the model assumes that the operator is in a region of "near-recall," where given a little more time, he might remember. A second

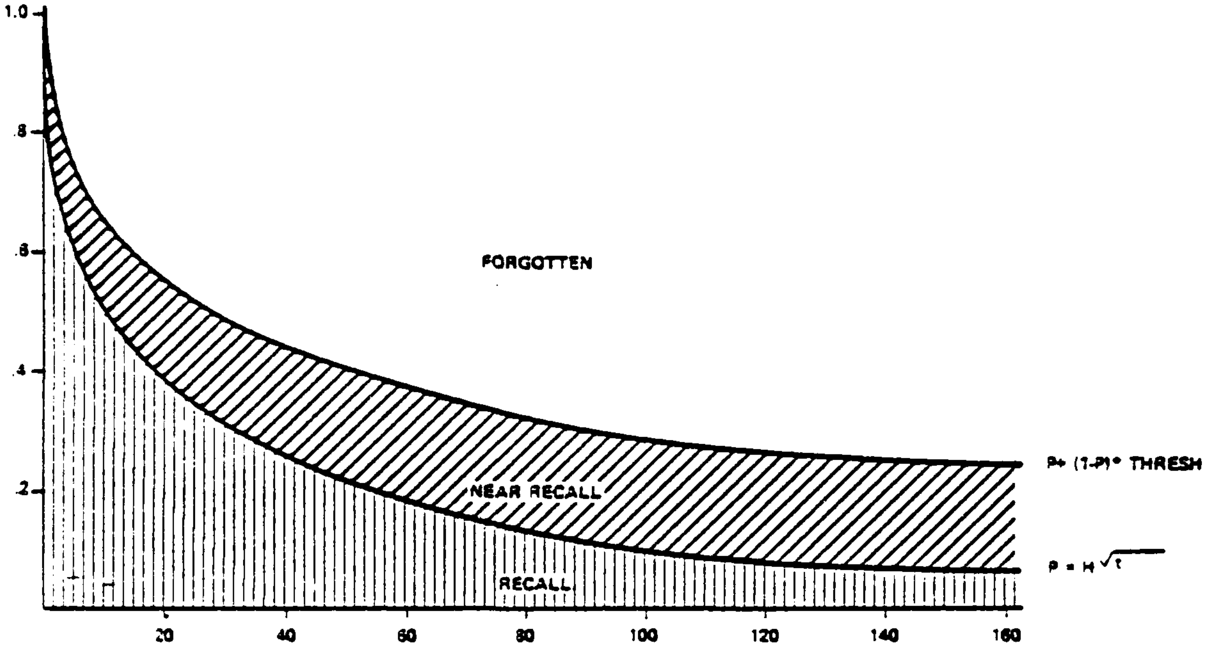


Figure 16. Short term recall regions.

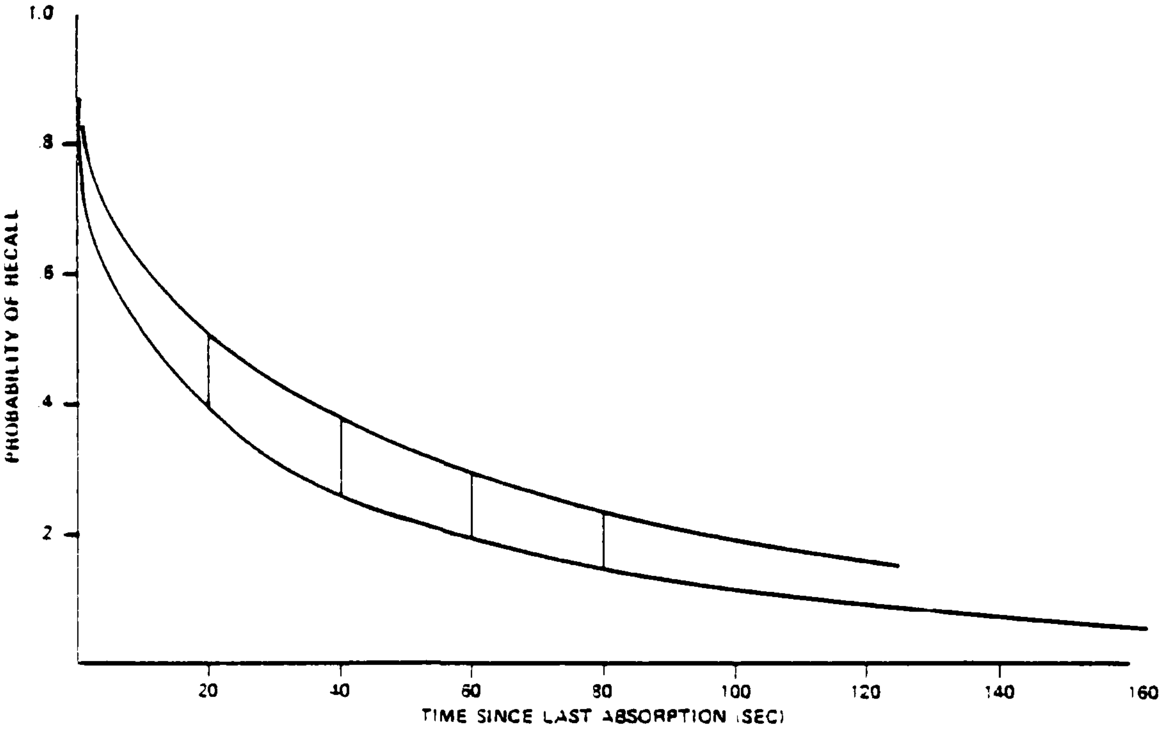


Figure 17. Recall increments for continuous devices.

random number is therefore drawn and compared with P to determine whether the information is remembered, forgotten, or in the near-recall region. Usually a second draw will suffice -- the random number will either be in the remembered or forgotten region. But the process could theoretically go on for three or more tries. Each try results in the addition of a small amount of time, the *short-term memory cycle time*, to the total time for retrieval from short-term memory (Figure 16).

When the operator recalls a value, the hab strength associated with that value is changed in order to simulate the effects of *rehearsal*. The remembered value is given a hab strength that is lower than if the information had been absorbed again, but higher than it would have been had the normal decay curve been followed (Figure 17).

There are several features of this recall model that deserve some comment (and probably some future work). First, the process by which the hab strength associated with any item of information is increased and recalled is independent of the value of information to the operator -- the threshold value is the same for all information and consequently all items of information follow essentially the same curves for the increase and decrease in hab strength. This is clearly unrealistic -- information that is of greater value to the operator should decay less rapidly and should be learned to a higher level of confidence than less important information. Secondly, the recall model has no explicit provision for allowing information to transfer from short-term memory to long-term memory, though there is an effective transfer that results from rehearsal for the real human operator. Third, there is no linkage between items of information -- if, for example, the operator depresses a switch that changes the value of a display, that action will normally not affect the value that the simulated operator will recall for the display, whereas a true operator would certainly know that the displayed value had changed.* And fourth, there are no external cues

*Although the analyst can, in fact, specify that such linkages exist when coding the procedures.

AREAS FOR IMPROVEMENT IN MEMORY MODEL

-- HAB STRENGTH IS INDEPENDENT OF THE <u>VALUE</u> OF INFORMATION

-- NO EXPLICIT PROVISION FOR TRANSFER FROM SHORT-TERM TO LONG-TERM MEMORY

-- NO LINKAGE BETWEEN ITEMS OF INFORMATION

-- NO IMPACT FROM EXTERNAL CUES

that impact the perceived or recalled value of a device, as the view out the window might cue the recall of the altimeter value for an aircraft pilot.

2.2.2 Errors During Recall

For continuous devices, there is a portion of the recall model that simulates the decreased accuracy associated with the recalled value. The basic premise behind this feature of the recall model is that as confidence (i.e., hab strength) in the value of a device decreases, the precision of the value that the operator recalls for the device will also decrease. Thus, if at some later time, the operator is asked for the value of the device, then the operator will be able to supply fewer "significant digits" as the time from the last absorption of the value of the device increases. We term this process *modular decay*. The modular decay function is such that given an initial device value of, for example, 123456 and an initial hab strength of .8, the modularly decayed values would be as shown in Figure 18.

2.2.3 Extrapolation of Values

If the operator can recall the value that a continuous device had the last time he read it, then HOS enables the operator to extrapolate its value to the current time. The extrapolation is linear and based on the two preceding absorbed values and the times when those values were obtained. It is the responsibility of the HOS user to declare whether or not extrapolation is to be permitted for each device.

2.2.4 Accessing the Information Recall Micro-Model

The analyst can force the operator to attempt to recall the value of a device by the use of the statement:

```
RECALL device
```

However, this statement is rarely used. Rather, the recall model is almost always accessed implicitly by simply including the device name within the

TIME SINCE ABSORPTION (T)	VALUE RECALLED IF RECALL SUCCEEDS	PROBABILITY THAT RECALL WILL SUCCEED (P)	TIME SINCE ABSORPTION (T)	VALUE RECALLED IF RECALL SUCCEEDS	PROBABILITY THAT RECALL WILL SUCCEED (P)
0	123456	1	0	123.456	1.0
1	123455	.8	1	123.455	.8
20	123000	.37	20	123.000	.37
40	125000	.24	40	125.000	.24
60	125000	.18	60	125.000	.18
80	125000	.14	80	125.000	.14
100	125000	.11	100	125.000	.11
120	125000	.09	120	125.000	.09
140	125000	.07	140	125.000	.07
160	125000	.06	160	125.000	.06

Figure 18. Modular decay examples.

context of another statement. When HOS recognizes that a device value is needed, it will attempt to recall the value.

2.4.5 Scope of the Information Absorption and Recall Models

The estimated value of a device is the only characteristic of a device that is either recalled or read by the HOS operator. The operator *does*, however, maintain other information on other device characteristics -- desired values, upper limits, lower limits, etc. -- but these quantities (termed device *parameters*) are considered to be resident in the operator's long-term memory and therefore are not subject to the information absorption/recall processes. The various device parameters are listed in Figure 19.

2.3 MENTAL COMPUTATION

The mental calculations performed by the HOS operator are termed *operator functions*, or simply *functions*.

The mental calculation micro-model uses the hab strength construct in much the same way as the information absorption and information recall micro-models. The result of a mental computation has an associated hab strength that represents the operator's confidence in the computed data. As the operator spends more time on the computation, his confidence in his estimate increases until either:

- The hab strength threshold is exceeded.
- The hab strength has asymptoted.
- The maximum number of iterations through the hab strength incrementing process has been exceeded.
- The amount of time spend in computation exceeds a maximum allowable computation time.

The recall model for mentally computed data is identical to the model used for any other type of data.

DEVICE PARAMETERS

- DESIRED VALUE
- RATE OF CHANGE
- TIME (OF LAST ESTIMATE)
- X AND Y COMPONENTS
- UPPER AND LOWER LIMITS
- CRITICALITY
- STATE (ACTIVE OR INACTIVE)
- ESTIMATED VALUE

Figure 19. Device parameters.

The basic difference between the mental computation model and the information absorption model is that, in the latter, information is absorbed from a display or control in the crewstation, whereas in the former, displayed information is used to determine a value that is not displayed anywhere in the crewstation. For example, a typical mental computation when driving an automobile is determining how much farther one can go on a tank of gas. The computation requires the absorption of an item of information (the amount of fuel remaining) coupled with some prior knowledge (the number of miles per gallon).

When a mental calculation is required, HOS will determine what information is needed in order to perform the calculation. If the HOS operator can remember the information, the calculation is performed at once. If he cannot remember the information, an appropriate sequence of actions is initiated to enable the operator to obtain the data. In the above example, the displayed information required is the amount of fuel remaining. If the operator cannot remember this, HOS would cause him to look at the fuel gauge and read its value.

Each mental calculation can require as many as ten different data items. These may be the values of displays or controls or the results of other mental calculations. An unlimited number of parametric values are also allowed. The amount of time required for a mental calculation is considered to be the amount of time required to gather all the items of information needed for the calculation plus some additional time to "put it all together." Because of the high potential variability in a function calculation, the analyst is required to supply a function computation time for each function -- HOS itself will supply the times required to gather all the items of information needed for the calculation.

A second difference between the mental computation and information absorption models is that in the information absorption model, the minimum hab strength associated with a device is dependent on the number of settings

MENTAL COMPUTATION

-- USES HAB STRENGTH CONSTRUCT

-- COMPUTATIONS REQUIRE INFORMATION FROM OTHER DEVICES

-- ANALYST SUPPLIES A COMPUTATION TIME

-- HAB STRENGTH IS DEPENDENT ON HAB STRENGTHS OF COMPONENT DATA

-- CALCUALTIONS ARE ERROR-FREE; DATA INPUT TO CALCULATIONS ARE NOT associated with the device. In the case of mental computations, the hab strength associated with the operator function is the minimum hab strength associated with any of the components in the function calculation.

Errors in mental computation are assumed to be the result of errors associated with the data that goes into the calculation itself. The calculation process itself is considered to be error-free. Thus, if the operator makes an error or obtains an inaccurate data value when either recalling the data or reading the data needed for a calculation, then the result of the calculation will be incorrect, or inaccurate, according to the incorrectness or inaccuracy of the incoming data. If the data values are correct and accurate, then the result of the calculation will be accurate. It should be noted, however, that, as a result of the way in which mental calculations are described to HOS, the analyst has the ability to inject errors into the function calculation if he so chooses.

2.4 MAKING A DECISION

HOS decision-making takes place at two levels -- the inter-procedural and the intra-procedural levels. To understand what is meant by this, we have to explain what is meant by a *procedure*, A procedure is an *operator task consisting of any number of steps*, any step of which can invoke the execution of another procedure or any other operator action. For example, the operator's *mission* in any particular simulation is a procedure that invokes other procedures -- a pilot's mission may invoke a procedure for takeoff, another for cruise, another for landing, etc. Within these procedures (or any procedures that they invoke) there are steps that describe operator actions -- reading a display, adjusting a control, etc. Decision-making can therefore operate at two different levels -- deciding what procedure to perform from a set of available *active* procedures, or deciding *what to do next* within any particular procedure.

HOS gives the analyst the option of both limited and total control over these decisions. The analyst can opt for *total control* in the sense that simulations can be constructed that *force* the operator to follow a specific sequence of steps and procedures. The analyst can, instead, choose *limited control* in the sense that the exact sequence of task and subtask operations that an operator will use is unknown -- the simulation can be constructed so that the HOS operator is allowed to make decisions for himself in accordance with a flexible task structure. Such a flexible structure is appropriate because, like a real operator, HOS can adapt its actions to situations. The power of HOS lies in its ability to adapt its performance to situations in a natural and realistic fashion.

Decisions about what to do next *within* a procedure are fairly simple. HOS will attempt to execute each step in a procedure in sequence until it can go no further, for whatever reason. If it finds itself blocked, it will attempt to "unblock" iteslf. If it can, it will continue marching forward; if it can't, it will look for some other procedure to work on, at which point the decision-making logic for selecting a procedure is invoked.

As it marches forward in a procedure, HOS may encounter a *statement* that requires a decision, i.e., an IF statement. The IF statement requires the operator to make a decision about the current status of information or events in the simulation. If the condition(s) tested is (are) satisfied, then it proscribes a set of actions to be taken. If the condition(s) is (are) not satisfied, the actions are not performed. A small time charge is assessed for this decision-making function over and above the time charges associated with gathering the information needed for the decision.

There are basically three types of events that will block the operator:

(1) An action is required that the operator cannot perform because the action requires body resources that are busy doing something else.

DECISION MAKING (WITHIN A PROCEDURE)

OPERATOR EXECUTES STEPS SEQUENTIALLY UNTIL "BLOCKED"

- BODY PART UNAVAILABLE
- CONTROL DEVICE UNAVAILABLE
- INFORMATION UNAVAILABLE

A STEP MAY REQUIRE A DECISION

ANOTHER PROCEDURE CAN BE INVOKED TO BE EXECUTED

- IMMEDIATELY
- AS TIME PERMITS
- PERIODICALLY

DECISION MAKING (BETWEEN PROCEDURES)

SELECTION OF A PROCEDURE IS DEPENDENT ON:

- CRITICALITY (PRIORITY) OF THE PROCEDURE
- HOW LONG IT HAS BEEN SINCE THE PROCEDURE WAS LAST EXECUTED

INITIAL CRITICALITIES SET BY ANALYST AND CAN BE CHANGED DYNAMICALLY

EFFECTIVE CRITICALITY FOR MONITOR PROCEDURES IS DEPENDENT ON HOW CLOSE A DEVICE IS TO ITS DEFINED LIMITS

(2) The operator requires information that is currently unavailable because a device in inactive (not enabled), or

(3) The operator must perform a control action that cannot be performed because the control is inactive (not enabled).

Of these situations, the latter two are the more common. When they occur, HOS will automatically invoke a special type of procedure -- an *enable* procedure -- whose function is to activate the device that is inactive. When the first situation occurs, HOS will simply go off and work on another procedure until the required body part is free.

One of the actions that can be performed within a procedure is the invocation of another procedure. When a procedure is invoked, the analyst can specify either that:

(1) The procedure is to be executed immediately and no more steps in the current procedure are to be executed until the invoked procedure has been completed, or

(2) The invoked procedure is to be placed on an *active procedure list* and is to be executed as soon as appropriate, or

(3) The invoked procedure is to be executed periodically until removed from the active procedure list.

In situation (1), control transfers immediately to the invoked procedure and no more steps in the invoking procedure are executed until the invoked procedure is completed. The active procedure list, formed by invoking procedures by methods (2) and (3), is the list of procedures available to the operator when he finds himself blocked in his current procedure. Procedures placed on the active procedure list by method (3) are called *monitor* procedures in that they are usually used to cause the operator to periodically monitor a particular display or control.

Finally, the analyst can force a procedure to be selected from the active procedure by using special forms of the IF and GO TO statements (see below).

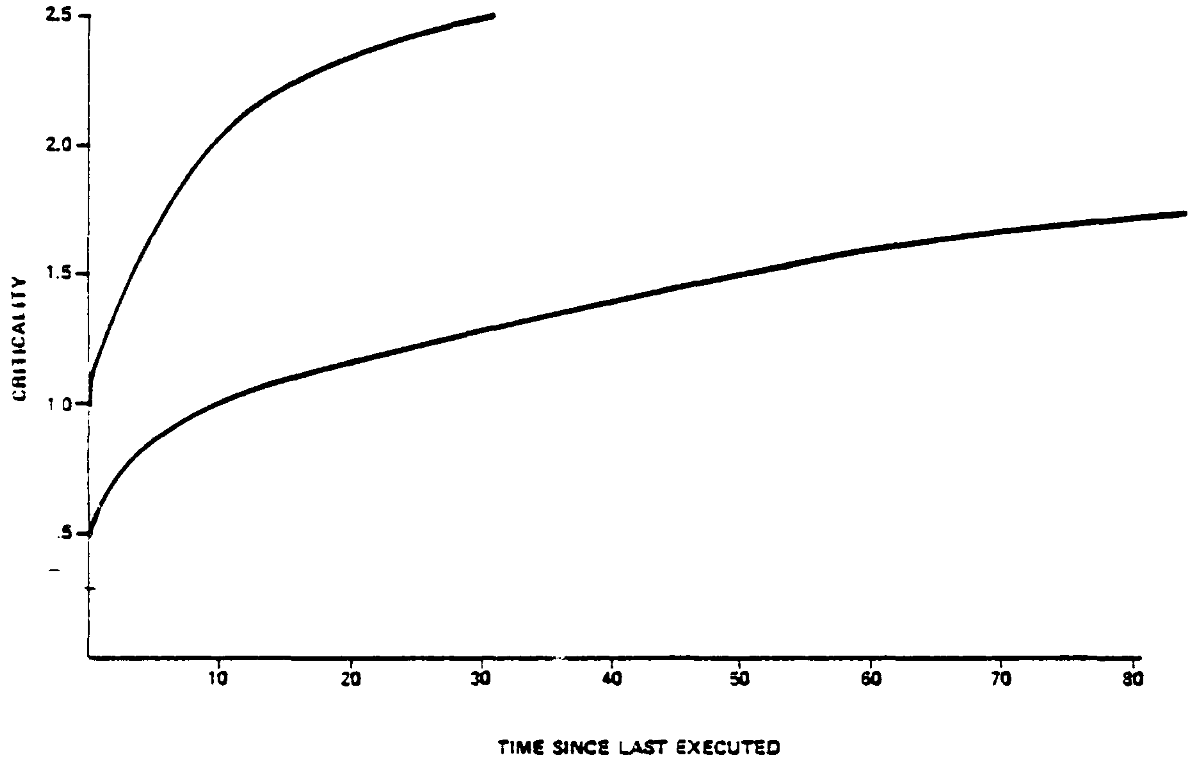


Figure 20. Increase in procedural criticality with time.

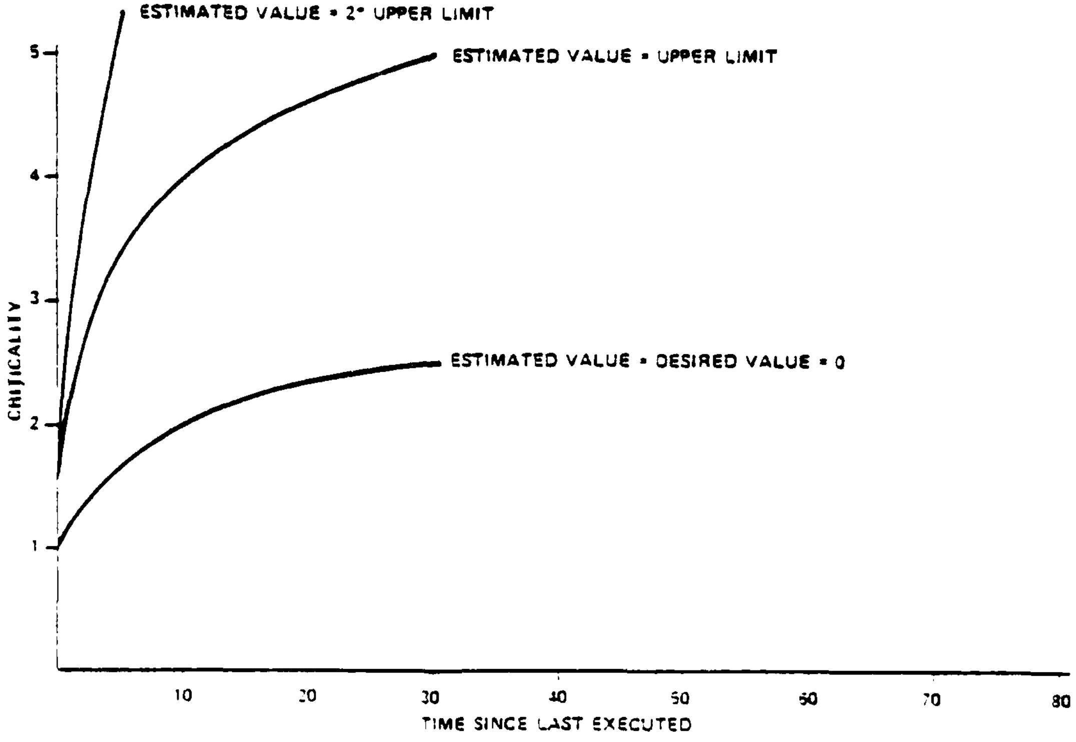


Figure 21. Increase in criticality for monitor procedures.

When a procedure is to be selected from the active procedure list, there is a model that represents the operator's procedural selection process. This model considers two factors:

(1) The criticality (priority) of the procedure, and

(2) How long it has been since the procedure was last executed.

A detailed discussion of the interaction of these factors is presented in Appendix A of this document. Briefly, as the length of time since the procedure was last executed increases, the *effective criticality* of the procedure increases (over an initial criticality that can be set by the analyst), as shown in Figure 20. In addition, for monitor procedures, the effective criticality is further modified by a factor that is dependent on how close the device being monitored is to a defined set of limits. As the estimated value of the device approaches its limits, the effective criticality of the device increases. When it exceeds the defined limits, the effective criticality increases very rapidly, as shown in Figure 21. The computed effective criticalities for each procedure on the active procedure list are compared and the procedure with the highest effective criticality is chosen as the next procedure to work on.

2.5 ANATOMY MOVEMENT

The anatomy movement micro-model is almost always accessed implicitly -- i.e., the analyst will rarely issue a command that will force a body movement. Rather, HOS itself will determine whether a body movement is required in order to accomplish the objective of an instruction. If it decides that a body movement is required, HOS will automatically select the appropriate body part, move it to the required location, and add to the simulation time a computed estimate of the amount of time the action would have taken a real operator. For example, suppose a procedural statement says:

TURN SWITCH-A ON.

BODY PART SELECTION

- EYES
- HANDS
  - -- RIGHT OR LEFT?
  - -- IS HAND BUSY?
  - -- SWAP HANDS?
- FEET
  - -- RIGHT OR LEFT?
  - -- IS FOOT BUSY?

If HOS decides that this action is necessary,* and if one of the operator's hands is not already on SWITCH-A,** HOS will select a hand, "move" it to SWITCH-A, and charge an amount of time equal to the time that a real operator would have taken to move that hand to SWITCH-A from wherever the hand was at the time the instruction was issued.

Thus, the moving and grasping primitive function consists of two micro-models -- one to determine which body part to use for a particular action, the other to assign a time charge for the movement. The body part selection micro-model is based on several common-sense principles. The first is that the body part to be used is determined by the function to be performed and the device being referenced. Thus, if the operator is going to be reading data from a device, the eyes are usually the appropriate body part to use. However, there may be some devices whose value cannot be determined visually -- touching them with a hand or foot may be more appropriate. Some devices may use two modalities -- the eyes are used to absorb information while the hands are used when the device is to be altered. HOS permits the analyst to specify for each device the most appropriate modality for each function (reading and/or altering).

If the operator's eyes are to be used for a specific function, there is no problem -- the HOS operator has only one pair of eyes which are immediately moved to the device. The time charge assigned for an eye movement is computed from an equation that was developed by fitting the data from an experiment that involved lateral eye movements (Dodge and Cline, 1901) and from an unpublished experiment by Wherry and Bittner that involved both lateral and convergence movements.

*SWITCH-A may be ON. A real operator, if he remembered this, would not perform the action. Similary, HOS would decide whether the simulated operator remembered whether the device was on and, if he did, would not initiate the body movement.

**Assuming that SWITCH-A is a device that is turned on by hand.

ANATOMY MOVEMENT (MOVING AND GRASPING)

- EXPLICIT ACCESS:

  LOOK AT THE ALTIMETER

- IMPLICIT ACCESS:

  TURN SWITCH-A ON

- SELECT APPROPRIATE BODY PART

- MOVE IT TO REQUIRED LOCATION AND ADD TIME

The equation:

$$T = .14324\ A + .0175$$

where

$$A = \max\ (\Delta\theta,\ \Delta\phi) + .2\ \min\ (\Delta\theta,\ \Delta\phi)$$

and

$$\Delta\phi = \left|\ \tan^{-1}\left(\frac{P_1}{1.275}\right) - \tan^{-1}\left(\frac{P_2}{1.275}\right)\ \right|$$

$$\Delta\theta = \left|\ \cos^{-1}\left(\frac{P_1\ P_2}{|P_1||P_2|}\right)\ \right|$$

P_1 = vector from design eye point to fixation point 1

P_2 = vector from design eye point to fixation point 2

assumes that both the lateral and convergence movements can proceed in parallel at the same rate with the total movement time being dependent on which movement takes the greater amount of time.

When one of the operator's hands is needed, the problem is not quite so simple -- it is necessary for HOS to decide which hand to use. The logic that HOS uses is fairly complex: (Figure 22.)

(1) HOS will attempt to use the hand that is currently closer to the device, unless that hand is currently busy doing something else.

(2) If the preferred hand is busy, but will be free "soon," where "soon" is an amount of time that can be set by the analyst, then HOS will "wait" until the preferred hand is free and will then "move" the operator's hand to the device.

(3) If the preferred hand will not be free soon, then the operator's *other* hand is used -- assuming that it is free and can reach the device.

(4) If the operator's other hand is *not* free, but will be soon, HOS will again wait until that hand is free and then use it.

- WHICH BODY PART IS APPROPRATE TO THE TASK
- WHICH BODY PART IS CLOSER
- IF THE PREFFERED BODY PART IS "BUSY," WILL IT BE FREE WITHIN A REASONABLE AMOUNT OF TIME
- IF IT WON'T BE FREE, CAN THE FUNCTIONS WHICH IT IS PERFORMING BE PERFORMED BY ANOTHER BODY PART

Figure 22. Anatomy Movement Logic

(5) If, however, the operator's other hand cannot reach the device, then a determination is made as to whether a *hand swap* should be initiated. In a hand swap, the less preferred hand takes over the function being performed by the preferred hand so that the operator can move the preferred hand to the device.

(6) If both hands are busy and won't be free for some time, or if a hand swap cannot be performed, HOS will decide that the instruction is unexecutable and will delay the execution of the procedure in which that statement is found until one of the operator's hands is free.

Similar logic pertains to the use of the operator's feet with the exception that "swaps" cannot take place.

The time required for a hand or foot movement is assumed to depend on both the magnitude and the precision of the movement. The equations that determine how long a hand movement will take are a combination of the results of experiments by Fitts and by Topmiller and Sharp and are discussed in detail in Appendix A of this document. These data are shown in Figure 23 where they are compared with other hand movement studies. The same equations are also currently being used for foot movements, but with different basic parameter values.

Some key characteristics of the anatomy movement micro-model that should be noted here are:

(1) Movements, like the instructions that initiate them, are executed serially for each body part.

(2) Movements are ballistic -- once initiated they cannot be stopped nor can another action be initiated while the movement is taking place.

(3) Movement times are fully deterministic, based on where a body part is and where it is being moved to -- there is no variability.

(4) If a movement cannot be performed, an interrupt will be generated enabling the operator to select another procedure from the active procedure list for execution.

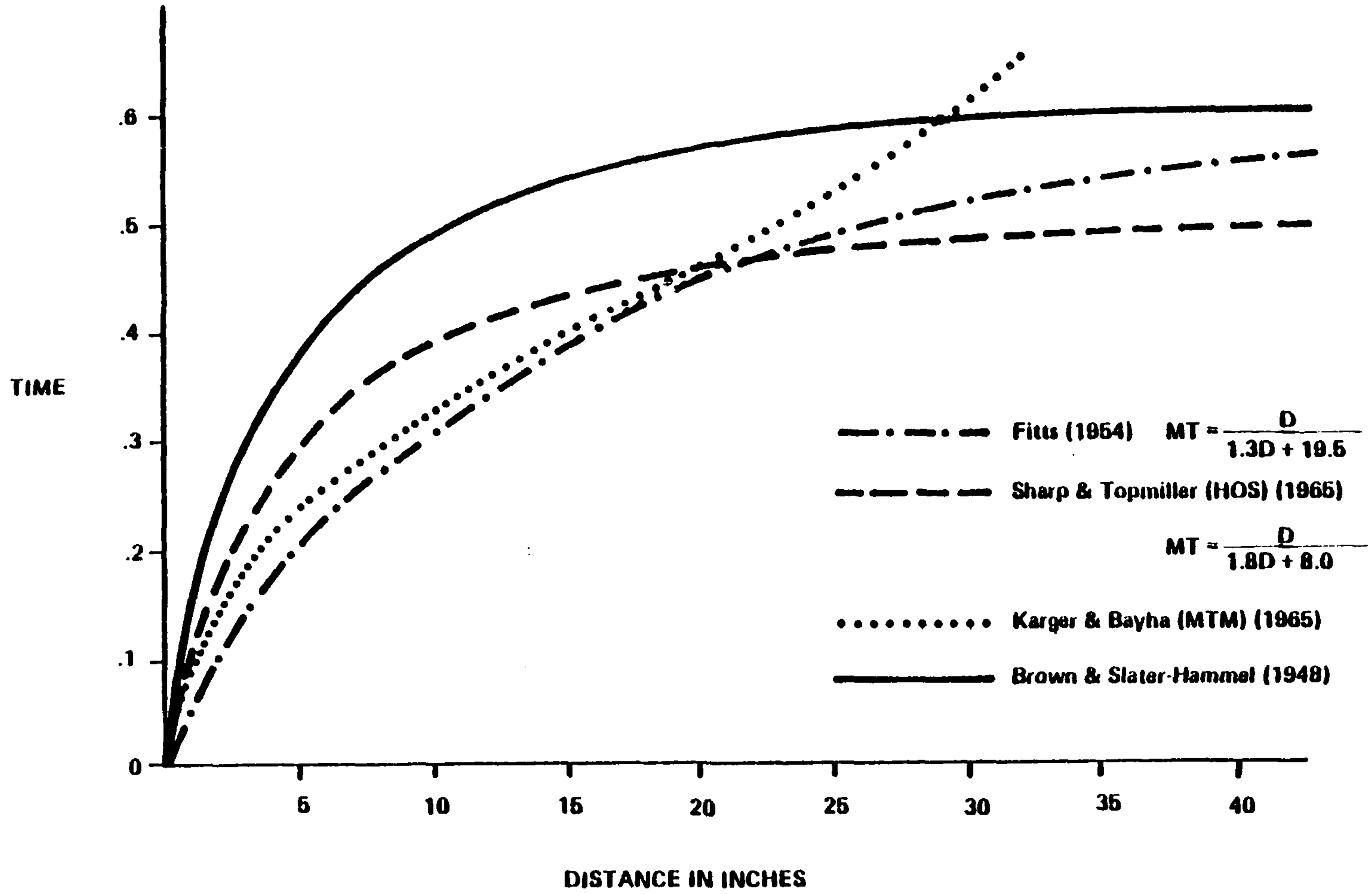


Figure 23. Hand movement times as a function of distance.

ANATOMY MOVEMENT MICRO-MODEL FEATURES

- ACTIONS ARE INITIATED SERIALLY FOR EACH BODY PART
- ACTIONS ARE BALLISTIC
- EACH BODY PART HAS A RELAXED LOCATION
- BODY PARTS RETURN TO RELAXED LOCATION AFTER A SPECIFIED TIME
- RETURN TO THE RELAXED LOCATION CAN BE OVERRIDEN BY SPECIFYING A "GRASP" LOCATION
- UNEXECUTABLE ACTIONS CAUSE SUSPENSION OF PROCEDURE

CONTROL MANIPULATION

DISCRETE CONTROLS

-- TIME TO MANIPULATE IS A FUNCTION OF THE NUMBER OF SETTINGS THAT WILL BE PASSED THROUGH

CONTINUOUS (ROTARY) CONTROLS

-- TIME IS DEPENDENT ON FORCE REQUIRED AND ANGULAR CHANGE

CONTROL MANIPULATIONS CAN BE PERFORMED IN PARALLEL

2.6 PERFORMING A CONTROL MANIPULATION

Times associated with control manipulations are highly variable because of the diverse types of controls used in different operator stations. Consequently, HOS allows the user to describe the characteristics of a control which are used to determine a set of equations that describe the time associated with a control manipulation. In addition, there are a set of "packaged" calculations that compute control manipulation times for two basic control types -- discrete controls and continuous rotary knobs.

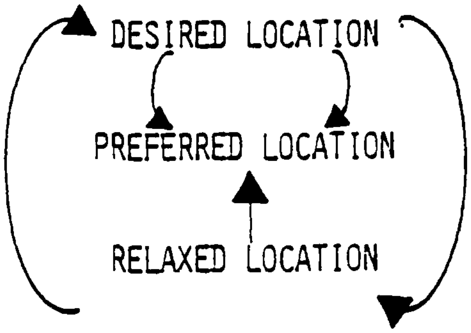
For discrete controls, the analyst is required to supply a time that represents the time required to move the ccntrol through a single setting. If a control manipulation results in a movement through several settings, the time assigned will be the time required for a single setting multiplied by the number of settings.

The formula for the manipulation time for a continuous rotary control was derived by fitting a quadratic to a table of data presented by Karger and Bayha (1966). The resultant formula is:

$$T = .0482 + .0050F + .0084\ FA$$

where F is the force in pounds required to turn the control and A is the angle through which the control is to be turned, in radians.

Unlike some of the other actions that we have discussed -- information absorption, recall, anatomy movement, etc. -- once initiated, control manipulations can proceed in parallel with other actions. Thus, the operator *can* be performing manipulations concurrently with both his right and left hands. In fact, one of the HOPROC lanugage constructs (the "parallel" alter) enables the analyst to specify that two or more actions *must* be carried out simultaneously.



BODY PARTS RETURN TO A RELAXED LOCATION WHEN NOT IN USE.

A "GRASP" LOCATION CAN BE ASSIGNED THAT TEMPORARILY OVERRIDES THE RELAXED LOCATION

Figure 24. Relaxation Logic

2.7 RELAXATION

The HOS relaxation micro-model interfaces with all the other action micro-models. Though fatigue itself is not currently modeled, HOS does exhibit one related characteristic that a real operator tends to exhibit -- when body parts are not busy doing anything else, the operator will move them to a comfortable, *relaxed location*. The analyst can override this default location by specifying a *grasp location* -- a location at which some action is expected. But after the operator has performed an action at the grasp location, the appropriate body part will automatically return to its relaxed location.

This logic is summarized in Figure 24. Any action establishes a location to which the operator must move in order to carry out the action. After the action has been carried out (and after a specified interval of time has elapsed) the body part will return to the grasp location (if one has been established) or to the relaxed location, if no other actions require that body part. After an action occurs at the grasp location, that location is eliminated, and body parts return to the relaxed location.

2.8 OPERATOR VARIABILITY

As described above, most of the equations that govern the operator micro-models in HOS are fully deterministic. This is consistent with two of the premises in the HOS model -- that the operator is a trained operator and that performance variations observed in experiments on individual operators are largely the result of situational differences, as opposed to differences in basic performance parameters. However, there are clearly differences in operator performance -- both between operators and for the same operator under different operational conditions. Some of the HOS operator parameters mentioned above enable the analyst to examine the effects of such differences -- the short-term memory cycle time, hab strength threshold, etc. In addition, by modifying the equations described above, one can readily describe an operator with a different performance profile.

OPERATOR VARIABILITY

-- EXPRESSED THROUGH OPERATOR PARAMETERS

- SHORT-TERM MEMORY CYCLE TIME
- HAB STRENGTH THRESHOLDS

⋮

-- CAN BE EFFECTED THROUGH PERFORMANCE EQUATIONS

-- "OPERATOR STATES" CONSTRUCT

Finally, there is a HOS construct that was a part of the original concept of HOS that was intended to model such performance differences under differing internal and external states. However, the *operator states* (o-states) concept has not as yet been implemented because of the challenge that has so far confronted us in modeling average performance when no special stresses are influencing the operator.

HOPROC

OBJECTIVE: A LANGUAGE THAT WAS

-- FLEXIBLE
-- ADAPTABLE
-- NON-SPECIFIC TO A PARTICULAR CREWSTATION

OBJECTIVE ACHIEVED WITHIN THE CONSTRAINT THAT HOPROC HAD TO BE ABLE TO BE USED TO SIMULATE A SPECIFIC CREWSTATION.

EXAMPLES:

IN DECLARATIONS SECTION, NO DETAILS ABOUT DISPLAY/ CONTROL LOCATIONS OR MOST OPERATIONAL CHARACTERISTICS ARE NEEDED.

FUNCTIONS AND PROCEDURES ARE MODULAR AND CAN BE READILY MODIFIED FOR USE IN OTHER SIMULATIONS.

HOPROC IS FREE-FORMAT, ENGLISH/FORTRAN-LIKE, AND CAN BE READILY UNDERSTOOD BY THOSE WITH LITTLE FAMILIARITY WITH IT.

3. THE HOPROC LANGUAGE

3.1 INTRODUCTION

The HOS mission description is written in the HOPROC language. The primary objectives in developing HOPROC have been to construct a language that would be flexible, adaptable, easy to understand, and non-specific to a particular crewstation, and that would, at the same time, be able to describe the operator's crewstation and tasks in sufficient detail that they could be accurately simulated. Attainment of these goals was constrained by the fact that a *specific* crewstation *must* be described in order for HOS to simulate the operator's performance within the crewstation. But, even with this constraint, we feel that HOPROC has achieved its major objectives.

Some of the ways in which these objectives have been met are:

(1) The actual locations of the displays and controls and their operational characteristics are not described in HOPROC because they are irrelevant to how the operator *uses* the equipment. These data are supplied at the time of simulation and can be readily modified in order to simulate different crewstation configurations.

(2) The descriptions of hardware and operator procedures and functions are highly modular so that the procedures and functions developed for one crewstation can be readily adapted for use in other simulations.

(3) HOPROC is a free-format, English/FORTRAN-like language that can be readily interpreted even by those with little familiarity with the language or the specific problem being simulated.

THE HUMAN OPERATOR PROCEDURES (HOPROC) LANGUAGE

HOPROC -- AN ENGLISH/FORTRAN-LIKE LANGUAGE USED TO DESCRIBE A CREWSTATION AND A MISSION TO HOS.

HOPROC HAS THREE MAJOR SECTIONS:

- A TITLE DECLARATIONS SECTION
- A FUNCTIONS SECTION
- A PROCEDURES SECTION

The HOPROC mission description has three major components:

(1) Title Declarations

(2) Function Definitions

(3) Procedure Definitions

The title declarations assign names to the various devices in the operator's crewstation. In addition, they describe certain general characteristics of the operator's displays and controls and the symbols that may appear on the operator's display screens. The primary device characteristics defined in the title declarations are the settings or scale factors associated with the devices. These device characteristics declarations enable HOS to examine the statements entered by the analyst in the functions and procedures definitions. HOS can then determine whether the analyst's description of the crewstation and of the operator's tasks is complete and consistent. If the description is not complete and/or consistent, HOS will indicate to the analyst where the crewstation and/or task descriptions are lacking.

The heart of the mission description is the procedure definitions. These describe the tasks that the operator must perform in order to carry out his mission, the effects on the hardware of any actions taken by the operator, and the independent external events that may occur during the simulation and that may impact on the operator's mission.

The function definitions describe both the mental calculations that the operator must perform in order to carry out the mission tasks and the mathematical calculations that occur during the hardware processing.

The title declarations, functions, and procedures are defined in a HOPROC data deck consisting of ten *sections*, which must be entered in the order shown in Figure 25. However, when developing a HOS simulation, one can rarely write the code for each of the sections in order from beginning to end. When working with an existing system, one will usually begin

SETTING SECTION	TITLE DECLARATIONS
OSTATE SECTION	
ARGUMENT SECTION	
DISPLAY SECTION	
CONTROL SECTION	
SYMBOL SECTION	
OPERATOR FUNCTIONS	FUNCTION DEFINITIONS
HARDWARE FUNCTIONS	
HARDWARE PROCEDURES	PROCEDURE DEFINITIONS
OPERATOR PROCEDURES	

Figure 25. The Sections in a HOPROC Data Deck

by identifying all the displays, controls, and symbols and defining their characteristics in the appropriate sections. Then, the names of all the settings can be collected into the SETTING SECTION associated with the displays, controls, and symbols.

The final step is to define the operator functions and procedures and the hardware functions and procedures. Usually, the operator sections can be developed independently from the hardware sections.

With a system that is still being planned, the same general sequence of HOPROC code development can be followed, but the progression is less likely to be as clear cut.

Therefore, in developing a simulation, there is a tendency to jump from section to section. As we introduce the HOPROC language to you in the following paragraphs, there will be a similar tendency. This may be somewhat confusing because, for example, variations on certain of the procedural statements will be introduced at different times in the discussion. In order to minimize the potential confusion, we have indicated in Figure 26 where each of the concepts discussed in the following sections are introduced. This index can also help you to locate the relevant portions of the discussions in both this manual and in the HOS Users' Guide, whenever you have a question about HOPROC syntax.

Since this section is only an introduction to HOPROC, all the options available for every HOPROC statement type will *not* be described. Complete details on the syntax of each HOPROC statement are presented in Volume II of the HOS documentation (the HOS Users' Guide) and on the HOS Reference Card. The purpose of this section is to provide novice HCS users with an introduction to the HOPROC language so that these references can be used more readily. Later sections will describe how one runs HOS and how its output is to be interpreted -- subjects that are not covered in any detail in Volume II.

To be supplied later.

Figure 26. Index

In the following sections, some paragraphs are marked with the letter A in parentheses (A). These paragraphs are for those wishing to learn some of the advanced features in HOPROC. It is recommended that the beginning student skip these sections, since they are not relevant to a basic understanding of HOPROC or of the radar plotting simulation.

3.2 TITLE DECLARATIONS

3.2.1 Displays and Controls

The operator's equipment complement is divided into two major functional groups -- *displays* and *controls*.* By a *display* we mean a device that conveys a *single item of information to the operator.* A *control* is a device that the operator uses to enter a *single item of information into the system* or to control a single function. These definitions obviously encompass a wide variety of very different devices -- gauges, lights, levers, pushbuttons, etc. When a simulation is run, HOS requires that explicit details about the characteristics of each display and control be supplied. But in HOPROC, every device in the operator's crewstation need only be identified as either a display or a control.

HOPROC ignores the fact that the actual displays and controls in the system may be integrated displays and multifunction controls. One of the potential uses of HOS is the examination of the effects of regrouping displays or controls or of changing control characteristics or the displayed information. Therefore, every item of information that can appear on an integrated display must be identified as a separate display. Similarly, every function that a multifunction control can perform must be called a separate control. When the simulation is run, display and control locations must be specified and it is at this time the the devices can be described as co-located.

*Symbols, which are a special type of display are described in Sections 3.2.9 through 3.2.11.

DISPLAYS

- CONVEY A SINGLE ITEM OF INFORMATION
- HAVE A FIXED LOCATION

CONTROLS

- USED TO CONTROL A SINGLE FUNCTION
- HAVE A FIXED LOCATION

A final characteristic of a HOS display or control is that, disregarding the movement of pointers and such, displays and controls do not move, relative to the operator.

The primary display used in the P-3C SS-3 radar plotting simulation is the multipurpose digital display, which we shall refer to as the RADAR-DISPLAY. The controls that will be used in the simulation are the LOAD switch, the TRACK-BALL, and three of the momentary contact switches on the keyset tray -- the RADAR-MODE pushbutton, the ENTER-RADAR-CONTACT pushbutton, and the HOOK-VERIFY pushbutton (Figure 27).

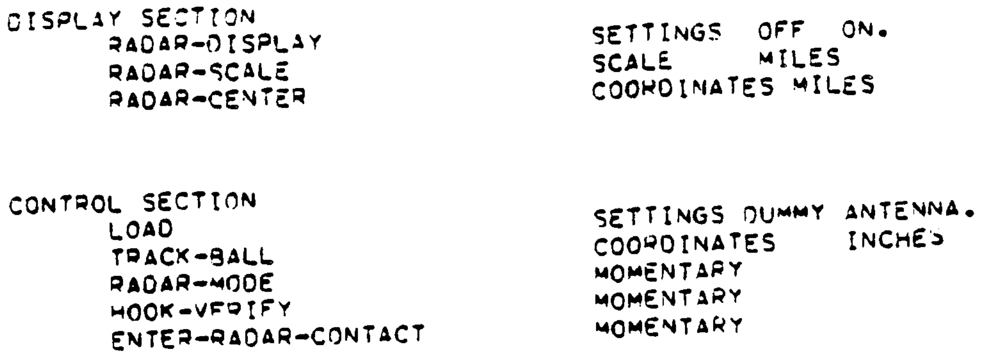
The words RADAR-DISPLAY, TRACK-BALL, etc., are termed *HOPROC variables*. Every display and control in the operator's crewstation must be given a *unique* variable name. Names can be of any length, but they must be unique within the first 20 characters. Names that consist of several English words, such as RADAR-DISPLAY and TRACK-BALL, must either use a hyphen to connect the individual words, or else the entire title must be enclosed in quotation marks so that HOS can recognize that the words comprise a single title.

3.2.2 The DISPLAY and CONTROL SECTIONs

HOPROC title declarations are organized so that all the displays are listed consecutively, followed by all the controls. Each of these sections is introduced by a card that identifies the section as either the DISPLAY SECTION or the CONTROL SECTION.

3.2.3 (A) Overriding the Section Declarations

Displays can be defined within the CONTROL SECTION by entering the keyword DISPLAY after the display title. Similarly, controls can be defined within the DISPLAY SECTION by entering the word CONTROL after the name of the control.



```
DISPLAY SECTION
        RADAR-DISPLAY                          SETTINGS  OFF  ON.
        RADAR-SCALE                            SCALE      MILES
        RADAR-CENTER                           COORDINATES MILES

 CONTROL SECTION
        LOAD                                   SETTINGS DUMMY ANTENNA.
        TRACK-BALL                             COORDINATES    INCHES
        RADAR-MODE                             MOMENTARY
        HOOK-VERIFY                            MOMENTARY
        ENTER-RADAR-CONTACT                    MOMENTARY
```

Figure 27. Display and control sections for the radar plotting simulation.

3.2.4 Types of Displays and Controls

Displays and controls can be either *discrete*, *continuous*, or *positional*.

Discrete devices are devices that have *settings*; continuous devices have *values* that are continuous over a defined range and that may have a *scale factor* associated with them. Using the example of an automobile, the gearshift for an automobile with a manual transmission is a discrete device with settings of either neutral, reverse, first, second, third (and, perhaps, fourth or fifth). The speedometer, on the other hand, is a continuous device whose value can range from 0 mph up to the maximum speed of the automobile.

Positional devices are a special type of continuous device. Specifically, positional devices are devices that have a vector value -- i.e., that have an *X* and *Y* value simultaneously (or a range and a bearing, magnitude and direction, etc.). An example of a positional device is the track-ball that is used to control the position of the cursor on the screen. Its value at any time can be represented by two numbers that express the *X* and *Y* components (or the magnitude and direction) of the displacement of the track-ball from an initial position.

3.2.5 Defining a Discrete Device

When a discrete display or control is being defined, the names of any settings associated with the display or control must be identified. This is done by following the name of the device with the word SETTINGS and the list of settings. The list of settings is terminated by a period. For example, in the radar plotting problem, the LOAD switch has two settings, DUMMY and ANTENNA. Therefore, the full definition of the LOAD switch is

LOAD SETTINGS DUMMY ANTENNA.

AD-A094 353　ANALYTICS INC　WILLOW GROVE PA　F/G 5/8
THE HUMAN OPERATOR SIMULATOR. VOLUME IX. HOS STUDY GUIDE.(U)
SEP 78　M I STRIEB, F A GLENN, R J WHERRY　N62269-78-M-6684
UNCLASSIFIED　TR-1320-VOL-9　NL

2 OF 4
AD A
094353

DEVICE TYPES

DISCRETE -- HAVE SETTINGS

CONTINUOUS -- HAVE VALUES THAT ARE CONTINUOUS OVER A RANGE

-- MAY HAVE AN ASSOCIATED SCALE FACTOR

POSITIONAL -- HAVE VECTOR VALUES

-- MUST HAVE AN ASSOCIATED SCALE FACTOR

The order in which the settings are listed is critical for controls since HOS uses the sequence of settings to determine the amount of time that the manipulation of a discrete control will take. Therefore, the sequence of settings must correspond to the order in which the operator passes through the settings when manipulating the control. For example, if the rotary switch shown in Figure 28 had been in the SS-3 operator's crewstation, the definition of the control would have been:

```
MODE-SELECTOR SETTINGS ON-LINE, ANALOG-TEST, MATRIX-TEST,
                       REGISTRATION-TEST, VECTOR-TEST, TYPE-
                       TEST, FUNCTION-GENERATOR-TEST, OFF-
                       LINE/ANALOG.*
```

For discrete displays, the order of the settings has no effect on the course of the simulation.

3.2.6 (A) Omission of Setting Titles

The sequence in which the setting titles appear is not critical for displays. In fact, the list of potential settings that a display may have may be omitted using the keyword ANY after the keyword SETTINGS. Use of the keyword ANY tells HOS that the display may have any legitimate setting title. However, use of this option is not recommended because it inhibits some of the error checks that HOS normally performs. Consequently, when the ANY option is used, there is an increased chance of making an error elsewhere in the simulation that will not be detected by HOS.

3.2.7 Momentary Contact Switches

Momentary contact switches are a special type of discrete control. Depression of a momentary contact switch generally activates the function performed by the control; a second depression deactivates the function. Consequently, although a momentary contact switch is a discrete control,

*The comma is optional punctuation that can be used in any HOPROC statement to improve readability.

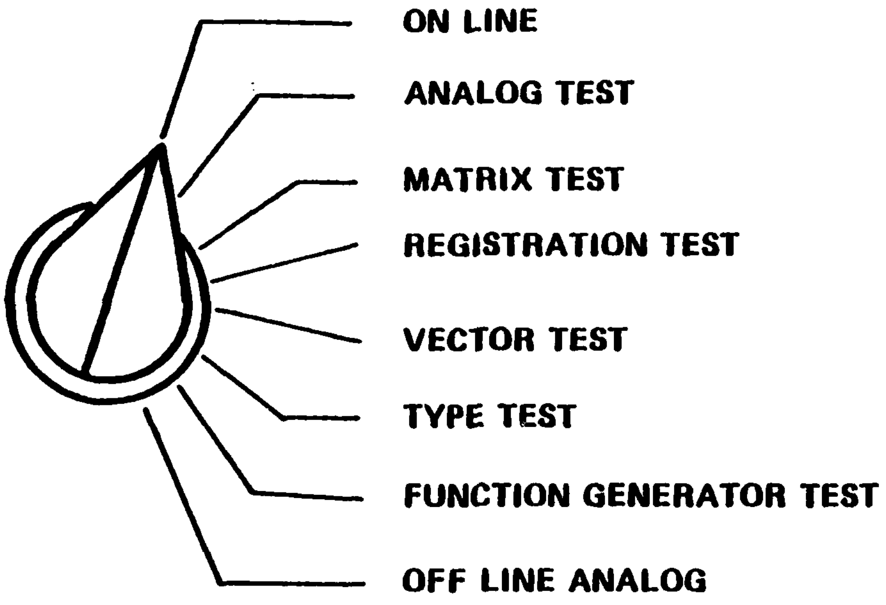


Figure 28. Example of multi-position discrete control.

it doesn't have settings, as such. The analyst can identify controls to HOS as momentary contact switches by substituting the keyword MOMENTARY for the keyword SETTINGS and the setting list. For example, in the radar plotting problem, the RADAR-MODE, ENTER-RADAR-CONTACT, and HOOK-VERIFY controls are all momentary contact switches. Consequently, they are defined by the statements:

```
RADAR-MODE MOMENTARY

ENTER-RADAR-CONTACT MOMENTARY

HOOK-VERIFY-MOMENTARY
```

3.2.8 Defining Continuous and Positional Devices

Continuous devices are devices that can have continuous values over a defined range. Continuous devices may, optionally, have a scale factor associated with them. The scale factor is entered after the keyword SCALE, which follows the device title.

For example, in the SS-3 crewstation, one of the items of information displayed on the radar display is the area covered by the display. Since this information is displayed in miles, it could be defined as:

```
RADAR-SCALE         SCALE MILES
```

If a continuous device does not have an associated scale factor, the word SCALE and the scale factor are omitted.

Positional devices *must* have scale factors. In order to identify them as positional, the word COORDINATES is used, instead of the word SCALE. For example, the TRACK-BALL is a positional device. Consequently, its definition is:

```
TRACK-BALL          COORDINATES INCHES
```

SYMBOLS

- CHARACTERISTICS CONVEY MORE THAN A SINGLE ITEM OF INFORMATION
- LOCATION CAN CHANGE THROUGH TIME
- MUST HAVE AT LEAST TWO CHARACTERISTICS
  - -- EXISTENCE
  - -- LOCATION

3.2.9 Symbols

Symbols are a special type of display that differ from standard displays in two significant ways. First, unlike standard displays, any particular symbol may convey *more* than a single item of information at a time. For example, the presence of a symbol on the RADAR-DISPLAY indicates the existence of an object such as a ship or aircraft in the real world. The shape of the symbol may indicate whether it is a ship or an aircraft. The symbol's position on the screen indicates the location of the object in the real world. If the real world object is moving, then the position of the symbol will be moving and its rate of movement will indicate how fast the real world object is moving. Thus, one of the characteristics of a symbol is that it can convey *more than a single piece of information at a time*. These items of information are termed the symbol's *characteristics*. Every symbol *must* have at least two characteristics -- *existence* and *position*.

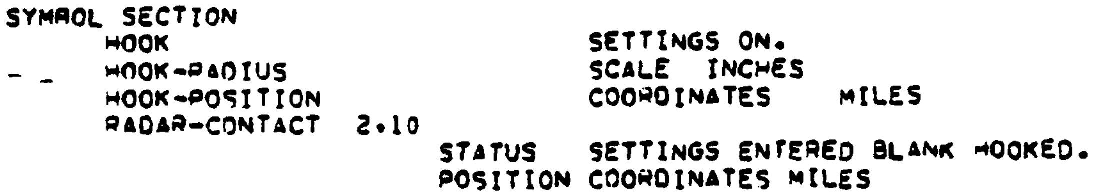
The second distinguishing feature of a symbol is that, unlike displays and controls, *the position of a symbol, relative to the operator, can change over time*. For example, the position of the symbol representing a moving ship on the RADAR-DISPLAY will change over time. As the position of the symbol moves, so too do the locations of all the symbol's characteristics. Displays and controls, on the other hand, must always remain fixed, relative to the operator.

3.2.10 The SYMBOL SECTION

Symbols are entered in a separate section of the HOPROC data deck. The symbol definitions are introduced by a SYMBOL SECTION card. The SYMBOL SECTION must follow the CONTROL SECTION. The SYMBOL SECTION for the radar plotting problem is shown in Figure 29.

3.2.11 Ordering of Symbol Characteristic Titles

HOS generally does not care about the order in which the names of the various displays, controls, and symbols in the crewstation are defined. However, because of the fact that each symbol has several



```
SYMBOL SECTION
      HOOK                                SETTINGS ON.
- -   HOOK-RADIUS                         SCALE  INCHES
      HOOK-POSITION                       COORDINATES    MILES
      RADAR-CONTACT  2.10
                               STATUS     SETTINGS ENTERED BLANK HOOKED.
                               POSITION COORDINATES MILES
```

Figure 29. Symbol section for the radar plotting simulation.

characteristics, all the characteristics for a symbol must be defined in a group -- they cannot be scattered throughout the SYMBOL SECTION.

Every symbol has two required characteristics -- its *existence* and its *position*. These two characteristics must be the first and last characteristics defined for each symbol. Any number of additional symbol characteristics may be defined between these two characteristics. The title that represents the symbol's existence must be discrete and must have at least one setting. The device title that represents the symbol's position must be defined with the word COORDINATES and a scale factor. For example, the HOOK in the radar-plotting problem is defined as follows:

```
HOOK        SETTINGS      ON.
HOOK-RADIUS       SCALE       INCHES
HOOK-POSITION COORDINATES MILES
```

Here, the title HOOK defines the existence of the hook on the display screen.* HOOK-POSITION refers to the real-world location that corresponds to the hook's location on the screen. Since real-world coordinates are measured in miles, the hook's coordinates are measured in miles.

The definition of the HOOK given above includes an additional characteristic of the HOOK that will be important in the radar plotting problem -- its radius (HOOK-RADIUS). Whenever the operator wishes to hook a symbol, he must manipulate the track-ball so that the symbol is encircled by the HOOK. In order to be able to determine whether a symbol is encircled, it is necessary to know the hook's radius. This distance is represented by HOOK-RADIUS.

*Since the HOOK must always remain on the screen, its only allowable setting is ON.

The HOOK-RADIUS has been defined with a scale of INCHES because it will be compared with the physical locations of other symbols on the screen. It could have been defined with a scale of MILES recognizing the fact that a HOOK-RADIUS of a particular size corresponds to a specific number of miles, depending on the scale to which the screen is set. We chose to define the HOOK-RADIUS as we did because of the fact that the number of miles represented by the hook-radius will vary with the display scale, whereas the physical size of the hook (in inches) will not change when the display is rescaled.

As another indication of the generality of HOPROC, notice that it is not necessary to specify what the actual value of the HOOK-RADIUS is in the HOPROC code. The actual value need only be specified when the simulation is run. The HOOK-RADIUS may be a small value, such that the operator will have to position the hook very precisely, or it could be a larger value that will not require such precise positioning. This is a design decision that will affect the operator's performance and HOS can be used to predict what the consequences of such decisions will be.

3.2.12 (A) Overriding Section Declarations

Displays and controls can be defined in the SYMBOL SECTION by following the display or control title by the keyword DISPLAY or CONTROL. Similarly, symbols can be defined in the DISPLAY or CONTROL sections by entering the keyword SYMBOL after the titles of *each of the symbol's characteristics.*

3.2.13 Device Groups

Often there are groups of devices that logically belong together and share similar names. For example, in a multi-engine aircraft, there are groups of displays and controls that are identical except for the fact that they are associated with different engines. The radar contact symbols on the SS-3 operator's radar display also have this property.

Rather than requiring the separate definition of each display, control, or symbol in a group as a device with its own unique set of characteristics, HOPROC provides a shorthand method to define such groups. First, the *group title* is entered, e.g.,

```
RADAR-CONTACT
```

Next, the number of individual displays, controls, and/or symbols or symbol characteristics associated with each *element* in the group is entered. For example, there are two characteristics associated with each radar contact -- its STATUS and its POSITION. Consequently, the number of symbol characteristics associated with each element in the group is two:

```
RADAR-CONTACT 2
```

This number (the number of *subgroups*) is followed by a number that represents the number of elements in the group. Thus, if a maximum of 10 radar contacts can appear on the screen at any one time, the number 10 would be entered after the number of subgroups:

```
RADAR CONTACT 2,10
```

Each of the subgroup titles is then defined as if it referred to an individual display, control, and/or symbol:

```
RADAR-CONTACT 2,10
STATUS    SETTINGS-   ON    OFF    ENTERED    HOOKED.
POSITION COORDINATES MILES
```

HOS will automatically generate titles for each subgroup and each element in the group. The titles will be formed by concatenating

DEVICE GROUPS

USED TO IDENTIFY LOGICALLY GROUPED DEVICES THAT SHARE SIMILAR NAMES.

- DURING THE SIMULATION, HOS WILL AUTOMATICALLY REFERENCE THE DESIGNATED ELEMENT (i.e., combining) the group title with the subgroup title and with the element numbers. For example, the titles that will be formed for the group defined above will be:

```
RADAR-CONTACT-STATUS
RADAR-CONTACT-1-STATUS
         :
RADAR-CONTACT-10-STATUS
RADAR-CONTACT-POSITION
RADAR-CONTACT-1-POSITION
         :
RADAR-CONTACT-10-POSITION
```

During the simulation, the analyst can refer to the subgroup titles without specifying a particular element number -- e.g.,

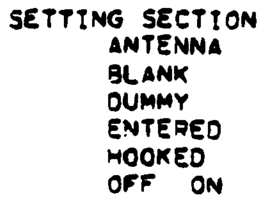
READ THE RADAR-CONTACT-POSITION.

HOS will automatically perform the appropriate actions on the specific element that has been determined to be "of interest" at that particular point in the simulation.

Note that the subgroups within the RADAR-CONTACT group are symbols defined within a group definition. As with other symbols, the first symbol characteristic must be discrete and the last must be positional. In this case, no additional characteristics were defined.

3.2.14 The SETTING SECTION

As the settings associated with each display, control, or symbol are determined, the setting titles must be collected into a SETTING SECTION. The SETTING SECTION is entered prior to the DISPLAY, CONTROL, and SYMBOL SECTIONS. Even though several different displays, controls, and symbols may share the same setting title, the title need be entered only once in



```
SETTING SECTION
        ANTENNA
        BLANK
        DUMMY
        ENTERED
        HOOKED
        OFF   ON
```

Figure 30. Setting section for the radar plotting simulation.

the SETTING SECTION. The SETTING SECTION for the SS-3 radar plotting problem is shown in Figure 30.

3.2.15 (A) Ordinals

Often, a display, control, or symbol will have settings that are numeric values. For example, the VHF channel selector on a television is a control that has the settings 2 through 13 and UHF. If such a control was being defined, the ORDINAL keyword could be used to eliminate the need to list all the numeric values in the SETTING SECTION and in the list of device settings. Using the keyword ORDINAL instead of the word SETTINGS in the device definition tells HOS that the device has a sequential list of numeric settings. For example, defining the channel selector as:

```
CHANNEL-SELECTOR ORDINAL 2 TO 13, UHF.
```

is equivalent to defining it as:

```
CHANNEL-SELECTOR SETTINGS 2,3,4,5,6,7,8,9,10,11,12,13, UHF.
```

and including the settings 2,3,4,5,6,7,8,9,10,11,12,13, and UHF in the SETTING SECTION. In the former case, only the setting UHF needs to be listed in the SETTING SECTION.

3.3 THE PROCEDURE DEFINITIONS

The preceding sections have described the major features of the HOPROC title declarations. In an actual HOPROC data deck, the functions are defined immediately after the title declarations. However, in *creating* a HOS simulation, it is usually much more natural to develop the operator and hardware procedures before creating the operator and hardware functions. Therefore, we will describe the features of the procedure definitions before discussing the functions.

```
            DEFINE THE MISSION.
                PERFORM RADAR-PLOT.
                END.

****************************************************************
            DEFINE THE PROCEDURE TO RADAR-PLOT.
                ENABLE THE RADAR-DISPLAY.
                IF ANY RADAR-CONTACT-STATUS IS NOT ENTERED THEN
ENTER:              DESIGNATE IT AS THE RADAR-CONTACT OF INTEREST;
                    MOVE THE HOOK-POSITION TO THE
                      RADAR-CONTACT-POSITION;
                    DEPRESS HOOK-VERIFY;
                    DEPRESS ENTER-RADAR-CONTACT.
                IF ANOTHER RADAR-CONTACT-STATUS IS NOT ENTERED
                    THEN GO TO ENTER NOW.
                END.

****************************************************************
            DEFINE THE PROCEDURE TO ENABLE THE RADAR-DISPLAY.
                TURN LOAD TO ANTENNA.
                END.

****************************************************************
            DEFINE THE PROCEDURE TO ADJUST THE HOOK-POSITION.
CHECK:              READ THE HOOK-POSITION.
                IF IT IS OK THEN END.
                DETERMINE THE TRACK-BALL-POSITION.
                MOVE THE TRACK-BALL TO THE RESULT.
                IF THE RATE OF THE TRACK-BALL IS NOT 0.0 INCHES
                    THEN WAIT.
                GO TO CHECK NOW.

****************************************************************
            DEFINE THE PROCEDURE TO ENABLE HOOK-VERIFY.

                ADJUST THE HOOK-POSITION.
                END.

****************************************************************
            DEFINE THE PROCEDURE TO ENABLE ENTER-RADAR-CONTACT.
                DEPRESS RADAR-MODE.
```

Figure 31. Operator procedures for the radar plotting simulation.

There are two parts to the procedure definitions. One part describes the *operator procedures* -- the tasks to be performed by the operator. The second part describes the *hardware procedures* -- the hardware consequences of the actions taken by the operator and the independent behavior of other systems being simulated. The rules governing statement syntax are almost identical in both sections. Therefore, in the discussion to follow, the characteristics of the operator procedures will be described first. Differences between the operator and hardware procedures will then be described.

3.4 THE OPERATOR PROCEDURES

The operator procedures for the radar plotting problem are shown in Figure 31. The operator procedures are introduced by an OPERATOR PROCEDURES statement. This statement is followed by the procedures themselves. Each procedure is introduced by a DEFINE statement and continues until the next DEFINE statement. Procedures can be defined in any order.

3.4.1 The DEFINE Statement

The first statement in the radar plotting operator procedures is:

```
DEFINE THE MISSION.
```

This statement is an example of a HOPROC DEFINE statement. The DEFINE statement begins with the word DEFINE and ends with a period. In between, there must be a procedure name, chosen by the analyst. In this case, the name MISSION was chosen as the name of the procedure.

The use of the name MISSION as the name of this procedure has a special significance to HOS -- specifically, it announces to HOS that this is the procedure that will control the simulation. When the simulation begins, HOS looks for a procedure named MISSION in the set of operator procedures. If it finds one, that procedure is used as the controlling procedure. If HOS doesn't find a procedure named MISSION, then it assumes

DEFINE STATEMENT

INTRODUCES A PROCEDURE DEFINITION.

PROCEDURES CAN BE DEFINED IN ANY ORDER.

MISSION SHOULD BE THE FIRST OPERATOR PROCEDURE.

THE DEFINE STATEMENT IS NON-EXECUTABLE -- I.E., IT DOES NOT RESULT IN ANY OPERATOR ACTIONS.

PERFORM STATEMENT

- PLACES THE NAMED PROCEDURE ON THE ACTIVE PROCEDURE LIST
- BEGINS THE EXECUTION OF THE NAMED PROCEDURE
- INHIBITS THE CONTINUATION OF THE CURRENT PROCEDURE UNTIL THE NAMED PROCEDURE IS COMPLETED

ACCOMPLISH IS A SYNONYM FOR PERFORM.

PERFORM VERB CAN BE OMITTED.

that the first operator procedure is the main procedure, no matter what its name is.

The word THE in the DEFINE statement is an example of a disregarded word. These are words that are completely ignored by HOS, but can be used to improve the readability of the procedure. The list of disregarded words is shown in Figure 32.

3.4.2 The PERFORM Statement

The first *executable* statement in PROCEDURE MISSION is an example of a PERFORM statement. The PERFORM statement identifies a procedure, defined elsewhere in the procedures section that must be executed before the current procedure can be continued. In this case, the statement:

```
PERFORM RADAR-PLOT.
```

says that the HOS operator is not allowed to continue with the MISSION procedure until the RADAR-PLOT procedure has been completed.

The word PERFORM is actually unnecessary. The instruction would have been understood equally well by HOS if the PERFORM had been omitted. In that case, the instruction would have been simply:

```
RADAR-PLOT.
```

The word ACCOMPLISH can be used as a synonym for the word PERFORM and any disregarded words can be inserted in the statement without changing its meaning. Thus, the statements:

```
ACCOMPLISH RADAR-PLOT.
```

A

AN

AT

IN

OF

THE

TO

ROUTINE

SUBROUTINE

PROCEDURE

Figure 32. Disregarded Words

and

ACCOMPLISH THE PROCEDURE TO RADAR-PLOT.

mean exactly the same thing as:

PERFORM RADAR-PLOT.

3.4.3 (A) The START Statement

If it is not necessary for the operator to complete a procedure before going on to the next statement, then the word START (or one of its synonyms -- COMMENCE, BEGIN, INITIATE, or ACTIVATE) is used instead of the word PERFORM. For example:

START RADAR-PLOT.

would tell HOS to put the radar-plotting procedure on the list of active procedures and to execute it as time is available.

3.4.4 (A) The COMPLETE Statement

If a particular procedure has been placed on the active procedure list by a START statement and at a later point it must be completely executed before continuing with the current procedure, then the verb COMPLETE can be used. For example, the statement:

COMPLETE RADAR-PLOT.

says that, if the RADAR-PLOT procedure is on the active procedures list, then it must be completed before going on to the next statement. If the procedure is not on the active procedures list, the COMPLETE instruction will be ignored.

START STATEMENT

- PLACES THE NAMED PROCEDURE ON THE ACTIVE PROCEDURE LIST
- SYNONYMS -- COMMENCE, BEGIN, INITIATE, ACTIVATE

COMPLETE STATEMENT

- FORCES THE NAMED PROCEDURE TO BE COMPLETED BEFORE THE CURRENT PROCEDURE CAN BE CONTINUED

END STATEMENT

- REMOVES THE NAMED PROCEDURE FROM THE ACTIVE PROCEDURE LIST
- IF THERE IS NO NAMED PROCEDURE, THE CURRENT PROCEDURE IS UNDERSTOOD

3.4.5 The END Statement

The second statement in MISSION is an example of an END statement. END statements are used whenever execution of a procedure is to be terminated. The basic format for the statement is:

END procedure-name.

If the name of the procedure is omitted, as in the example, HOS automatically understands that the procedure to be terminated is the procedure currently being defined -- in this case MISSION. Since the END statement is the last statement in MISSION, it could also have been omitted entirely. HOS would have automatically known that when the RADAR-PLOT procedure had been completed, the MISSION was over.

3.4.6 The ENABLE Procedures

The RADAR-PLOT procedure referenced in MISSION is shown in Figure 33. The first statement in this procedure is:

ENABLE THE RADAR-DISPLAY.

This statement is actually a special form of the PERFORM statement. It says that there is a procedure named ENABLE THE RADAR-DISPLAY that is to be executed before the RADAR-PLOT procedure can be continued. The function of an ENABLE procedure is to *activate* a display or control so that the information presented on the display can be read by the operator or, in the case of a control, so that a control manipulation can be performed. As with a standard PERFORM statement, the ENA3LE procedure must be executed before the current procedure can be continued. However, if the display or control named in the ENABLE statement is already active, HOS will automatically ignore the ENABLE statement.

```
          DEFINE THE PROCEDURE TO RADAR-PLOT.
               ENABLE THE RADAR-DISPLAY.
               IF ANY RADAR-CONTACT-STATUS IS NOT ENTERED THEN
ENTER:              DESIGNATE IT AS THE RADAR-CONTACT OF INTEREST;
                    MOVE THE HOOK-POSITION TO THE
                      RADAR-CONTACT-POSITION;
                    DEPRESS HOOK-VERIFY;
                    DEPRESS ENTER-RADAR-CONTACT.
               IF ANOTHER RADAR-CONTACT-STATUS IS NOT ENTERED
                    THEN GO TO ENTER NOW.
               END.
```

Figure 33. The RADAR-PLOT procedure.

ENABLE PROCEDURES

-- "ACTIVATE" DISPLAYS OR CONTROLS
-- "ENABLE" INFORMATION TO BE READ FROM DISPLAYS OR CONTROL MANIPULATIONS TO BE PERFORMED

ENABLE STATEMENTS

-- SPECIAL FORM OF THE PERFORM STATEMENT
-- ARE IGNORED IF THE DISPLAY OR CONTROL IS ALREADY ACTIVE

ALTER STATEMENT FOR DISCRETE CONTROLS

-- CAUSES THE OPERATOR TO CHANGE THE SETTING OF A CONTROL TO A SPECIFIED SETTING

-- INITIATES RECALL, INFORMATION ABSORPTION, ANATOMY MOVEMENT, AND CONTROL MANIPULATION MICRO-MODELS

-- RESULTS IN THE EXECUTION OF A HARDWARE PROCEDURE ASSOCIATED WITH THE CONTROL

SYNONYMS FOR ALTER -- TURN, CHANGE, MODIFY, VARY, MANIPULATE, PUSH, PRESS, DEPRESS, PULL, TWIST, SET, ENGAGE, SWITCH, PLACE, MOVE, INCREASE, DECREASE

3.4.7 The ALTER Statement for Discrete Controls

The procedure to ENABLE THE RADAR-DISPLAY consists of a single statement:

TURN LOAD TO ANTENNA.

This statement is an example of an ALTER statement for a discrete control.* The statement tells the HOS operator to change the LOAD switch from its current setting to the setting ANTENNA. If the LOAD switch is already in the ANTENNA position, the HOS operator will ignore the instruction; if it is in the DUMMY position (as it will be at the beginning of the simulation), the operator will move his hand to the control and switch it to the desired setting, ANTENNA.

Changing the control's setting will have effects on the hardware which will be described later when the hardware procedures are discussed. For now, we will simply assume that changing the control's setting will activate the radar display, thereby enabling the operator to read the positions of the radar-contacts appearing on the display screen.

3.4.8 The IF ANY Statement

The next statement in RADAR-PLOT:

IF ANY RADAR-CONTACT-SYMBOL IS NOT ENTERED THEN ...

is an example of an IF statement. The general format for an IF statement is:

IF condition THEN statement(s).

*The words TURN, CHANGE, MODIFY, VARY, MANIPULATE, PUSH, PRESS, DEPRESS, PULL, TWIST, SET, ENGAGE, SWITCH, PLACE, MOVE, INCREASE, and DECREASE are all synonyms for ALTER.

IF ANY STATEMENT

CAUSES THE OPERATOR TO SEARCH THROUGH A GROUP OF DEVICES LOOKING FOR ONE THAT SATISFIES THE TEST CONDITIONS.

SELECTED ELEMENT WILL BE AUTOMATICALLY REFERENCED WITHIN THE CURRENT PROCEDURE OR ANY ENABLE PROCEDURE INVOKED BY THE CURRENT PROCEDURE.

In this case, the "condition" is:

ANY RADAR-CONTACT-SYMBOL IS NOT ENTERED

This is a complex type of condition referred to as an "IF ANY" condition. IF ANYs are used whenever the analyst wishes to search through a *group* of items, looking for a specific one that satisfies the test condition. In this particular case, the HOS operator is being told to search through the symbols in the RADAR-CONTACT group looking for one that has not, as yet, been entered. The HOS operator will look at each radar contact in turn, searching for the first contact that has not been entered. If one is found, he will perform the actions specified in the statements that follow the keyword THEN. If one is not found, the operator will ignore the statements in the THEN clause and continue execution at the first statement after the period that terminates the THEN clause.

3.4.9 The DESIGNATE Statement

If a radar contact symbol is found that has not been entered, HOS will execute the instructions in the THEN clause of the IF statement. The first statement following the keyword THEN:

DESIGNATE IT AS THE RADAR-CONTACT OF INTEREST;

is an example of a DESIGNATE statement. This statement "designates" the element in the RADAR-CONTACT group identified by the IF ANY as the RADAR-CONTACT "of interest," i.e., the element in the radar contact group to be used whenever any reference is made to the RADAR-CONTACT group.

The DESIGNATE statement can also be used to designate a *specific* element (without the use of an IF ANY). For example, if element three in

DESIGNATE STATEMENT

IDENTIFIES A SELECTED GROUP ELEMENT AS THE ELEMENT OF INTEREST.

PUNCTUATION

PERIODS -- TERMINATE A SENTENCE

SEMI-COLONS & AND -- CONNECT CLAUSES IN THE THEN PORTION OF AN IF STATEMENT

COMMAS -- ARE DISREGARDED

PARENTHESES CAN BE USED TO DELIMIT COMMENTS the RADAR-CONTACT group was to be used whenever any of the items in the group was referenced, the statement:

DESIGNATE 3 AS THE RADAR-CONTACT OF INTEREST.

could be used.

3.4.10 (A) Scope of an IF ANY

Within the procedure in which an IF ANY statement is used, any references to the group being searched by the IF ANY will automatically be understood to refer to the particular element that satisfied the IF ANY. In addition, any ENABLE procedure begun by the procedure that uses the IF ANY and that references the group will also be automatically assumed to be referring to the group element chosen by the IF ANY. However, in order to tell *other* procedures to use the particular element selected by the IF ANY, the element must be "designated" by a DESIGNATE statement.

3.4.11 Punctuation

Up to this point, every procedural statement that we have encountered has been terminated by a period. The statements in the THEN clause, however, are terminated by a semi-colon. The semi-colon is used when there are several statements in a THEN clause that are to be executed as a consequence of having satisfied the IF statement. *All* statements in a THEN clause connected by semi-colons (or by the conjunction AND) will be executed when the IF is satisfied and skipped whenever the IF is not satisfied.

3.4.12 The ALTER Statement for Displays and Symbols

The second statement in the THEN clause:

MOVE THE HOOK-POSITION TO THE RADAR-CONTACT-POSITION;

is another example of an ALTER statement. In this case, though, the device being altered, HOOK-POSITION, is a symbol characteristic rather than a

ALTER STATEMENT FOR DISPLAYS AND SYMBOLS

-- CHANGES THE DESIRED VALUE OF A DISPLAY OR SYMBOL TO A NEW VALUE.

-- PLACES THE ADJUST PROCEDURE FOR THE DISPLAY OR SYMBOL ON THE ACTIVE PROCEDURE LIST.

-- DOES NOT RESULT IN AN IMMEDIATE CHANGE TO THE ACTUAL VALUE OF THE DISPLAY OR SYMBOL.

control. Because HOOK-POSITION is a symbol characteristic, this ALTER statement is significantly different from the ALTER statement (TURN LOAD TO ANTENNA) that we encountered previously. This is because, unlike controls, displays and symbols cannot be altered *directly*. Rather, when the *desired* value of a display or symbol is to be changed, a procedure must be invoked that uses a control to change the actual value of the display or symbol.* The procedure that is invoked is termed an *adjust* procedure. Figure 34 is an example of such a procedure -- the procedure to ADJUST THE HOOK-POSITION. When the ALTER statement:

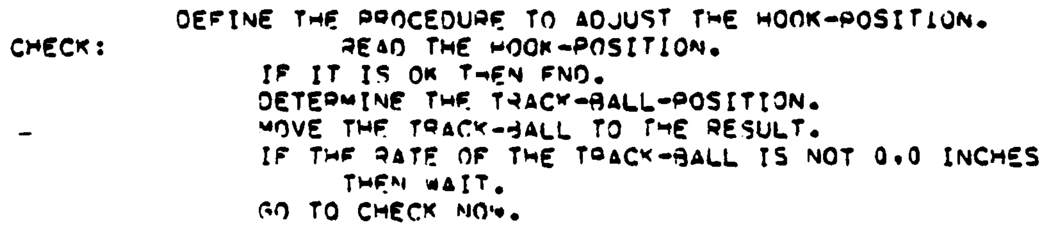
MOVE THE HOOK-POSITION TO THE RADAR-CONTACT-POSITION.

is encountered, this ADJUST procedure is placed on the active procedure list, to be executed when time is available. ADJUST procedures placed on the active procedure list by an ALTER statement are *not* executed immediately. Consequently, the hook will *not* move to the position of the radar contact immediately. Rather, it will be moved only when the operator has time available, *or* when some *other* instruction is executed that forces the procedure to be executed.

*Any of the device parameters (DESIRED, ESTIMATED, CRITICALITY, STATE, UPPER, LOWER, HAB-STRENGTH, TITLE, ACTUAL, DESIGNATED, RATE, TIME, XVALUE, and YVALUE) *can* be referenced in an ALTER statement, e.g.:

CHANGE THE UPPER (LIMIT) OF THE ALTIMETER TO 1000 FEET.

The current remarks apply *only* to the DESIRED value, which is the parameter most frequently referenced in the operator procedures and the parameter assumed by default when no parameter is specified for the device title that immediately follows the ALTER verb.



```
            DEFINE THE PROCEDURE TO ADJUST THE HOOK-POSITION.
CHECK:                READ THE HOOK-POSITION.
                IF IT IS OK THEN END.
                DETERMINE THE TRACK-BALL-POSITION.
                MOVE THE TRACK-BALL TO THE RESULT.
                IF THE RATE OF THE TRACK-BALL IS NOT 0.0 INCHES
                     THEN WAIT.
                GO TO CHECK NOW.
```

Figure 34. The procedure to ADJUST THE HOOK-POSITION.

3.4.13 The ALTER Statement for Momentary Contact Switches

Momentary contact switches are discrete controls that have no specific settings to which they can be set. Consequently, the ALTER statement for a momentary contact switch reduces to simply:

```
DEPRESS control.
```

where control is the name of the momentary contact switch. The next two statements in the RADAR-PLOT procedure:

```
DEPRESS HOOK-VERIFY;
```

and

```
DEPRESS ENTER-RADAR-CONTACT.
```

are examples of ALTER statements for momentary contact switches.

3.4.14 Implicit Invocation of ENABLE Procedures

The statements:

```
DEPRESS HOOK-VERIFY;
```

and

```
DEPRESS ENTER-RADAR-CONTACT.
```

share an interesting characteristic -- namely, that the actions *cannot* be performed because *certain prerequisites have not been satisfied.* In particular, in order for the operator to successfully perform the HOOK-VERIFY function, he must have first moved the hook to its desired position. ENTER-RADAR-CONTACT, on the other hand, cannot be depressed until after the HOOK-VERIFY function has been performed *and* until after the RADAR-MODE switch has been depressed.

IMPLICITLY INVOKED ENABLE PROCEDURES

-- EXECUTED IN ORDER TO ENABLE A CONTROL MANIPULATION TO BE PERFORMED.

-- DESCRIBE THE PREREQUISITES THAT MUST BE SATISFIED.

The ENABLE procedure for HOOK-VERIFY (Figure 35) ensures that the operator will position the hook over the radar contact before depressing HOOK-VERIFY. The ENABLE procedure for ENTER-RADAR-CONTACT (also shown in Figure 35) ensures that the operator will depress the RADAR-MODE, thereby activating the radar-matrix subfunctions, before depressing ENTER-RADAR-CONTACT. These procedures will be automatically invoked by HOS before each control is depressed. They are, therefore, said to be *implicitly invoked ENABLE procedures*.

The statements in the two ENABLE procedures are similar to some of the statements that have been discussed above. For example, the statement:

DEPRESS RADAR-MODE.

in ENABLE ENTER-RADAR-CONTACT is another example of an ALTER statement for a *discrete control and the statement:*

ADJUST THE HOOK-POSITION.

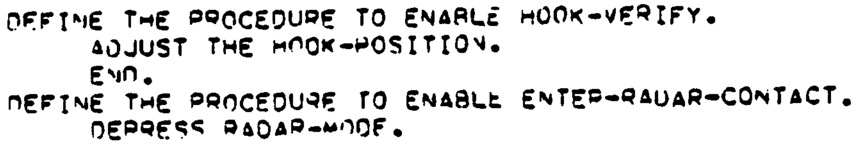
in ENABLE HOOK-VERIFY is another special form of the PERFORM statement. In the case of ADJUST THE HOOK-POSITION, however, the procedure to be executed, ADJUST THE HOOK-POSITION, had already been placed on the active procedure list by the statement:

MOVE THE HOOK-POSITION TO THE RADAR-CONTACT-POSITION.

The statement:

ADJUST THE HOOK-POSITION.

in ENABLE HOOK-VERIFY ensures that the procedure will be executed at once.



```
DEFINE THE PROCEDURE TO ENABLE HOOK-VERIFY.
      ADJUST THE HOOK-POSITION.
      END.
DEFINE THE PROCEDURE TO ENABLE ENTER-RADAR-CONTACT.
      DEPRESS RADAR-MODE.
```

Figure 35. The ENABLE procedure for HOOK-VERIFY and ENTER-RADAR-CONTACT.

3.4.15 (A) MONITOR and DISABLE Procedures

We have encountered two special types of operator procedures so far -- ENABLE procedures and ADJUST procedures. There are two other special types of operator procedures -- MONITOR procedures and DISABLE procedures. DISABLE procedures are simply the reverse of ENABLE procedures -- they deactivate devices. However, unlike ENABLE procedures, DISABLE procedures are not implicitly invoked by other statements.

MONITOR procedures are simply ADJUST procedures that are to be periodically executed in order to keep a display or control at some desired value. MONITOR procedures are placed on the active procedure list by a MONITOR statement. For example, the statement:

MONITOR THE ALTIMETER.

would place a procedure named MONITOR THE ALTIMETER (or ADJUST THE ALTIMETER) on the active procedure list to be periodically executed until removed from the active procedure list by an END statement.

3.4.16 The IF ANOTHER Statement

The IF ANY statement:

IF ANY RADAR-CONTACT-SYMBOL IS NOT ENTERED THEN ...

identified the first element in the RADAR-CONTACT group that satisfied the specified condition. The IF ANOTHER statement:

IF ANOTHER RADAR-CONTACT-SYMBOL IS NOT ENTERED THEN ...

continues the search, looking for the *next* element in the group that satisfies the test condition.

MONITOR PROCEDURES

-- ADJUST PROCEDURES THAT ARE TO BE EXECUTED PERIODICALLY

-- PLACED ON ACTIVE PROCEDURE LIST BY A MONITOR STATEMENT

DISABLE PROCEDURES

-- DEACTIVATE A DISPLAY, CONTROL, OR SYMBOL, THEREBY REQUIRING AN ENABLE PROCEDURE TO BE EXECUTED BEFORE THE DISPLAY OR SYMBOL CAN BE READ OR THE CONTROL MANIPULATED

-- EXECUTED THROUGH THE USE OF A DISABLE STATEMENT

3.4.17 The GO TO Statement

The GO TO statement transfers control to another statement elsewhere in the current procedure. The statement to which control is to be transferred must be identified by a *statement label*. The statement label is a HOPROC variable that precedes the statement with which it is associated. The label must be followed by a colon (:) to identify it as a label. For example, the GO TO statement in the RADAR-PLOT procedure:

GO TO ENTER.

transfers control to the statement identified by the label ENTER (the DESIGNATE statement). Transfers can only be made to labeled statements in the same procedure as the GO TO statement. GO TO statements cannot transfer to statements in *other* procedures.

Usually, when a transfer to another statement occurs, HOS assumes that a logically connected sequence of steps has been completed. Therefore, it will give other procedures on the active procedure list an opportunity to be executed before transferring to the labeled statement. However, often the analyst may not want the operator to have the option of working on other procedures. If this is the case, the analyst can say:

GO TO label NOW.

The keyword NOW tells HOS to transfer immediately to the labeled statement. No other procedures will be allowed to be executed before the transfer to the labeled statement has been completed.

3.4.18 The READ Statement

The first statement in the ADJUST HOOK-POSITION procedure:

READ THE HOOK-POSITION.

GO TO STATEMENTS

-- TRANSFERS CONTROL TO ANOTHER STATEMENT

-- TRANSFER MAY BE IMMEDIATE OR SUCH THAT OTHER PROCEDURES MAY INTERVENE

-- STATEMENT TO WHICH CONTROL IS TO BE TRANSFERRED MUST BE IN THE SAME PROCEDURE

READ STATEMENT

-- FORCES OPERATOR TO READ INFORMATION

-- SYNONYMS -- CALCULATE, COMPUTE

is an example of a READ statement. In the statements discussed so far, it has been assumed that the operator would automatically read or recall whatever information was needed, as it was needed. However, because these statements allowed the operator to recall information, the data could have been recalled incorrectly. Use of the READ statement forces the operator to read the specified information, rather than allowing him to rely on recall. Thus, it ensures that the operator will use the current value of the HOOK-POSITION, rather than a possibly incorrect remembered value in determining whether an adjustment is necessary.

3.4.19 The IF... OK Test

The statement:

MOVE THE HOOK-POSITION TO THE RADAR-CONTACT-POSITION.

in RADAR-PLOT established the RADAR-CONTACT-POSITION as the desired value for the HOOK-POSITION. The ADJUST procedure for the HOOK-POSITION describes the actions that must be performed to ensure that the HOOK has been moved to its desired position. One way to test whether the HOOK is at its desired value is to use an "IF... OK" test:

IF THE HOOK-POSITION IS OK THEN...

The OK test compares the current estimated value of HOOK-POSITION with upper and lower limits established around the desired value of the HOOK-POSITION and determines whether the estimated value is within those limits. If the estimated value is within the limits, then the test is satisfied. If the estimated value is not within the limits, then the test fails.*

*The OK test can also be specified as:

IF THE HOOK-POSITION IS WITHIN LIMITS THEN...

IF... OK STATEMENT

-- TESTS WHETHER A QUANTITY IS WITHIN A DEFINED SET OF LIMITS

-- LIMITS CAN BE ESTABLISHED BY A SET LIMITS... STATEMENT (A VARIANT FORM OF THE ALTER STATEMENT) OR IN A FUNCTION

-- EQUIVALENT TO

IF... IS WITHIN LIMITS THEN...

3.4.20 (A) Limits on Desired Values

The limiting range over which the estimated value can vary must be established in the HOPROC code, either within a procedural statement or as part of one of the operator or hardware functions, before the IF... OK instruction is executed. Within the operator procedures, the limits can be set by an ALTER statement, e.g., the statement:

SET LIMITS OF ALTIMETER TO 1000 FEET.

will set the upper and lower limits to 1000 feet on either side of the current desired value of the altimeter. The upper and lower limit values will change as the desired value of the altimeter is changed in order to maintain the specified relationship to the desired value. For example, if the desired value of the altimeter is 25000 feet at the time the SET LIMITS statement is encountered, then the upper and lower limits will be 26000 and 24000 feet respectively. If the desired value of the altimeter is changed to 30000 feet, the upper and lower limits will automatically be changed to 31000 and 29000 feet respectively, retaining the 1000 foot differential between the desired value and each of the limiting values.

In the SS-3 radar plotting example, the HOOK-POSITION limits have been defined within a hardware function (HOOK-LIMITS) rather than a procedural statement. This was done because the limiting values for the HOOK are dependent upon the display factor and the size of the hook radius, in a way that requires a potentially complex mathematical expression. Since expressing complex mathematical functions is much easier in HOPROC function statements than in HOPROC procedural statements, the limits were defined in a function rather than in a procedure.

3.4.21 (A) Use of the Pronoun IT

The pronoun IT has been used in both the DESIGNATE statement in RADAR-PLOT, and in the IF... OK statement in ADJUST THE HOOK-POSITION. In the DESIGNATE statement, the IT referred to the element in the RADAR-

THE COMPUTE STATEMENT

- INVOKES A FUNCTION CALCULATION

- THE RESULT OF THE FUNCTION CALCULATION IS THE HOPROC VARIABLE RESULT

CONTACT group that satisfied the IF ANY statement. In the IF... OK statement, the IT referred to HOOK-POSITION. In general, the pronoun IT refers to the first title following the introductory verb in the preceding statement. Thus, in the ADJUST THE HOOK-POSITION procedure, the statement preceding the IF IT IS OK... was:

READ THE HOOK-POSITION.

Since the title that followed the verb READ was HOOK-POSITION, this was the title that was being referred to by the pronoun IT. In the case of the DESIGNATE statement, the IT referred to the RADAR-CONTACT-SYMBOL that satisfied the IF ANY.

3.4.22 Invoking a Function Calculation -- The COMPUTE Statement

The next statement in the ADJUST THE HOOK-POSITION procedure:

DETERMINE THE TRACK-BALL-POSITION.

is an example of a COMPUTE statement.* The statement is similar to the READ statement discussed previously, except for the fact that it refers to a mental calculation that the operator must perform, rather than to an observable display, control, or symbol in the operator's crewstation. The mental calculation is the determination of the position to which the TRACK-BALL is to be moved in order to move the HOOK to its desired position. The details of this calculation will be discussed in Section 3.5.4. For now, we shall simply assume that the computation of the function will enable the operator to determine where the TRACK-BALL is to be moved. The next statement:

MOVE THE TRACK-BALL TO THE RESULT.

*DETERMINE is a synonym for COMPUTE, as are CALCULATE and READ.

is an ALTER statement for a control (TRACK-BALL). This statement causes the operator to initiate the TRACK-BALL manipulation. The keyword RESULT refers to the result of the TRACK-BALL-POSITION calculation.

3.4.23 Parameters, Positionals, Scale Factors, and Wait Conditions

Moving the HOOK to a specific point on the screen, which may itself be moving, is an example of a pursuit tracking problem. There are a variety of ways in which such problems have been modeled by those studying human operator performance. Through its flexible structure, HOS is capable of accommodating many of these models. For purposes of this discussion, however, we have chosen to let the basic structure of HOS and some of its constructs dictate the tracking model that we would implement. The model that we have chosen to use is a discrete model -- the operator reads the position of the hook, determines an amount by which he must move the track-ball in order to move the hook to its desired position, initiates the movement, and then waits until the movement is complete before deciding whether another movement is necessary. This final "action" -- waiting until the movement is complete -- is expressed by the next statement:

IF THE RATE OF THE TRACK-BALL IS NOT 0,0 INCHES THEN WAIT.

When the movement is complete, i.e., when the statement above is satisfied, then the statement:

GO TO CHECK NOW.

is executed. This statement recycles the operator through the statements that we have just discussed until the IF IT (i.e., the HOOK-POSITION) IS OK statement is satisfied, at which time the ADJUST procedure is terminated.

The statement:

IF THE RATE OF THE TRACK-BALL IS NOT 0,0 INCHES THEN WAIT.

has a number of interesting features that have not been encountered in any of the IF statements discussed so far. These features are:

- The explicit use of parameters.
- The use of positional values.
- The use of scale factors.
- The WAIT condition.

The following sections will deal with each of these features in detail.

3.4.24 (A) Explicit Use of Parameters

All of the statements that have been used so far have involved the *implicit* use of parameters. For example, HOS has automatically understood that when a device was referenced in an IF statement, its estimated value was to be understood. Thus, the other IF statement in ADJUST THE HOOK-POSITION could have read:

```
IF THE ESTIMATED VALUE OF THE HOOK-POSITION IS OK THEN END.
```

and the IF ANY statement in RADAR-PLOT could have been:

```
IF ANY ESTIMATED VALUE OF RADAR-CONTACT-STATUS IS NOT ENTERED
  THEN...
```

The IF statement:

```
IF THE RATE OF THE TRACK-BALL...
```

differs from these other IF statements because of the fact that it *explicitly* refers to a parameter associated with the TRACK-BALL, its RATE.

Whenever a parameter other than ESTIMATED, DESIRED, and STATE is referenced in an operator procedures statement, it must be explicitly specified. HOS will *generally* understand when the ESTIMATED, DESIRED, and STATE parameters are being referenced. However, it may be necessary at times for these parameters to be explicitly identified as well, when a non-standard usage is desired. The general rules regarding the use of parameters are as follows:

(1) In an IF statement, both the parameter associated with the variable immediately following the word IF, and the parameter associated with the variable after the conditional phrase are assumed to be ESTIMATED values, unless otherwise specified.

(2) In an ALTER statement, the parameter associated with the variable following the ALTER is assumed to be the DESIRED value; the parameter associated with the variable after the TO or BY is assumed to be ESTIMATED value, unless otherwise specified.

(3) If the keywords ACTIVE or INACTIVE are used in an IF statement or in an ALTER statement, the parameter STATE is automatically understood.

In the example, the parameter RATE had to be explicitly specified in order to override the ESTIMATED VALUE default parameter.

3.4.25 (A) Positional Quantities in IF and ALTER Statements

In some of the IF and ALTER statements that we have discussed, we have referenced positional variables, e.g., HOOK-POSITION and RADAR-CONTACT-POSITION in:

```
MOVE THE HOOK-POSITION TO THE RADAR-CONTACT-POSITION.
```

and HOOK-POSITION in:

```
IF THE HOOK-POSITION IS OK THEN END.
```

In these cases, HOS recognized the fact that the variables were positionals and automatically separated the values of the variables into their X and Y components. In the current IF statement:

IF THE RATE OF THE HOOK-POSITION...

we are referring to a parameter associated with the HOOK-POSITION, its rate of change (RATE), and we wish to test the value of the RATE against a specific numeric value. But since the HOOK-POSITION is a positional quantity, its rate-of-change is also positional.* Therefore, we must specify two numeric values against which the X and Y components, respectively, are to be tested. These two values are specified as 0,0 INCHES in the test condit'on.

3.4.26 (A) Use of Scale Factors in Procedural Statements

When devices defined with an associated scale factor are referenced in procedural statements, HOS will automatically check to make sure that the scale factors are compatible and, if necessary, will apply the appropriate conversion factors. Thus, if HOOK-POSITION and RADAR-CONTACT-POSITION had been defined with different (but compatible) scale factors, HOS would automatically have performed the appropriate conversions when the statement:

MOVE HOOK-POSITION TO THE RADAR-CONTACT-POSITION.

was encountered in the RADAR-PLOT procedure. When, as in the case of the IF statement:

IF THE RATE OF THE HOOK-POSITION IS NOT 0,0 INCHES THEN...

a numeric value is used as the test-condition, the units associated with the numeric value must be specified. If the units are incompatible with

*Only some of the parameters associated with a positional quantity are themselves positional -- these parameters are DESIRED, ESTIMATED, UPPER, LOWER, ACTUAL, and RATE.

OPERATOR FUNCTIONS

DESCRIBE THE OPERATOR'S MENTAL CALCULATIONS.

USES A HYBRID VERSION OF FORTRAN.

ENABLES THE VALUES OF DISPLAYS, CONTROLS, AND SYMBOLS TO BE REFERENCED AND COMBINED WITH "IMPLICIT" KNOWLEDGE.

the units associated with the device being referenced, or if no units are specified, HOS will issue an error message.

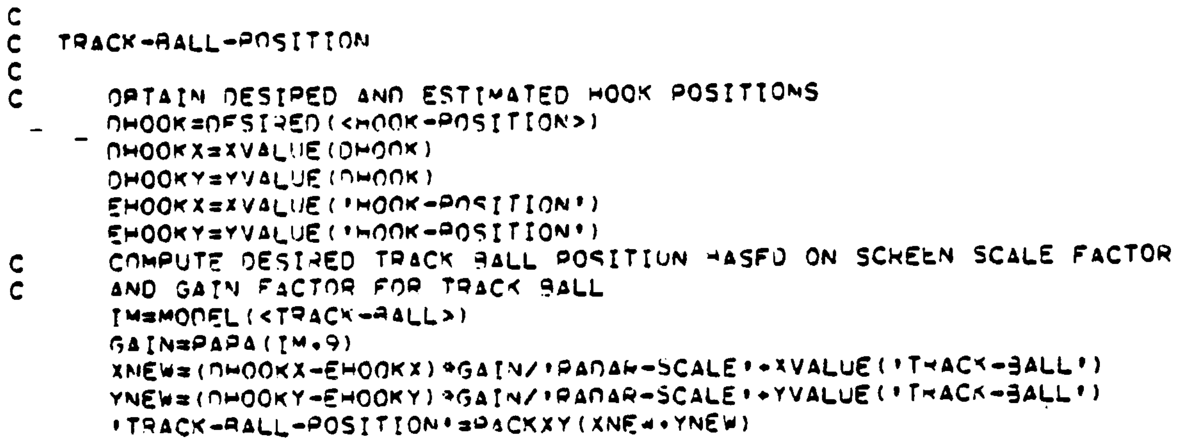
In the case of the RATE parameter, the numeric value should be in "units per second," (e.g., INCHES PER SECOND) rather than simply "units," since the RATE parameter is a time derivative. However, the HOPROC syntax processor does not recognize the phrase PER SECOND. Therefore, it should not be used.

3.4.27 (A) The WAIT Clause

Rather than specifying a statement or set of statements to be executed when the condition in an IF statement is satisfied, a WAIT clause can be used to indicate that nothing more is to be done in the current procedure until the condition being tested by the IF statement has been satisfied. The WAIT clause enables the HOS operator to work on other procedures while waiting for the condition to be satisfied. HOS will periodically check the condition and will continue execution of the procedure when the condition is satisfied.

3.5 OPERATOR FUNCTIONS

Operator functions describe the mental calculations that the operator is to perform using the information available from his displays and controls. In the radar plotting procedures, a single operator function, TRACK-BALL-POSITION, is used. This function calculates the position to which the operator wishes to move the track-ball in order to move the hook to its DESIRED value. This calculation requires the operator to combine an estimate of the current position of the HOOK with the desired position for the HOOK, and with some implicit knowledge of the display/control relationship between a TRACK-BALL movement and a change in the HOOK's position. Since descriptions of calculations like this can become extremely complex, it is not very efficient to use English-like statements to describe the calculation. Therefore, HOPROC uses a special hybrid version of one of the standard mathematical computer languages, FORTRAN, to express function



```
C
C  TRACK-BALL-POSITION
C
C      OBTAIN DESIRED AND ESTIMATED HOOK POSITIONS
       DHOOK=DESIRED(<HOOK-POSITION>)
       DHOOKX=XVALUE(DHOOK)
       DHOOKY=YVALUE(DHOOK)
       EHOOKX=XVALUE('HOOK-POSITION')
       EHOOKY=YVALUE('HOOK-POSITION')
C      COMPUTE DESIRED TRACK BALL POSITION BASED ON SCREEN SCALE FACTOR
C      AND GAIN FACTOR FOR TRACK BALL
       IM=MODEL(<TRACK-BALL>)
       GAIN=PARA(IM,9)
       XNEW=(DHOOKX-EHOOKX)*GAIN/'RADAR-SCALE'+XVALUE('TRACK-BALL')
       YNEW=(DHOOKY-EHOOKY)*GAIN/'RADAR-SCALE'+YVALUE('TRACK-BALL')
       'TRACK-BALL-POSITION'=PACKXY(XNEW,YNEW)
```

Figure 38. The TRACK-BALL-POSITION operator function.

calculations. This hybrid FORTRAN enables potentially complex mathematical relationships between HOPROC variables to be expressed concisely, while enabling the full computational power of FORTRAN to be used.

Figure 36 is an example of an operator function -- specifically, the operator function that defines the TRACK-BALL-POSITION calculation referenced in the operator procedure ADJUST THE HOOK-POSITION. If you are familiar with FORTRAN, you will notice the resemblance between the HOPROC-FORTRAN statements and standard FORTRAN. Even if you are not familiar with FORTRAN, you will notice that the HOPROC-FORTRAN statements are very similar to standard mathematical equations. In general, all the standard rules of FORTRAN apply to the coding of operator functions. There are some special characteristics of HOPROC-FORTRAN that make it different from standard FORTRAN. These differences are explained in the following sections.

3.5.1 Definition of a Function

There may be many operator functions defined in a single simulation althougn the radar plotting simulation uses only one. Each function ends with a line of code that includes the name of the function, enclosed in quotation marks, on the left-hand side of an equals sign. This line of code is the *only* line in which the name of the function (in quotation marks) can appear on the left-hand side of an equals sign.

3.5.2 Referencing Displays, Controls, and Symbols in a Function

Any HOPROC variable representing a display, control, or symbol can be referenced in either of two ways:

(1) By enclosing the HOPROC variable name in quotation marks, or

(2) By enclosing the HOPROC variable name in left and right carets (<and>).

OPERATOR FUNCTIONS

-- DEFINED BY THE NAME OF THE FUNCTION IN QUOTATION MARKS ON THE LEFT SIDE OF AN EQUALS (=) SIGN

-- CAN REFERENCE DISPLAYS, CONTROLS, SYMBOLS, OR OTHER FUNCTIONS

- ESTIMATED VALUES ARE REFERENCED BY ENCLOSING NAME IN QUOTATION MARKS
- OTHER PARAMETERS ARE REFERENCED BY ENCLOSING NAME IN CARETS

Variable names that are enclosed in quotation marks reference the ESTIMATED value of the display, control, or symbol. HOS will automatically access the necessary micro-models in order to either recall or absorb the ESTIMATED value. Thus, the two statements in the function TRACK-BALL-POSITION:

```
EHOOKX = XVALUE ('HOOK-POSITION')

EHOOKY = YVALUE ('HOOK-POSITION')
```

both reference the ESTIMATED value of the HOOK-POSITION. The statement:

```
XNEW = (DHOOKX-EHOOKX)*GAIN/'RADAR-SCALE' + XVALUE
       ('TRACK-BALL')
```

references the ESTIMATED values of both RADAR-SCALE and TRACK-BALL.

As many as 10 different ESTIMATED values can be referenced in a single function. Each ESTIMATED value can be referenced as many times as required (or desired) within the function. Thus, the two references to HOOK-POSITION in the statements above count as only a single reference towards the maximum of 10 ESTIMATED values allowed per function.

Variable names must be enclosed in carets and parentheses when any parameter *other* than the ESTIMATED value is to be referenced. For example, in the calculation of the TRACK-BALL-POSITION, when the DESIRED value of the HOOK-POSITION is to be referenced, the HOPROC variable name, HOOK-POSITION, must be enclosed in carets and parentheses after the parameter name, DESIRED, as in the statement:

```
DHOOK = DESIRED (<HOOK-POSITION>)
```

3.5.3 (A) Referencing a HOPROC Variable's Dictionary Entry Number

In certain cases, it is necessary to refer in a function to the *dictionary entry number* assigned by HOS to a HOPROC variable.* The dictionary entry number can be referenced by simply enclosing the HOPROC variable name in carets. One must, however, be careful when referencing certain types of HOPROC variables by their dictionary entry numbers and, in particular, when referencing grouped devices through their dictionary entry number. When referencing a display, control, or symbol group, subroutine REF must be called to replace the dictionary entry number of the subgroup with the dictionary entry number of the designated subgroup element before the dictionary entry number is used. For example, if we wanted to refer to the dictionary entry number of the subgroup element selected by the IF ANY statement:

```
IF ANY RADAR-CONTACT-STATUS IS NOT ENTERED THEN...
```

the following statements would be used:

```
ID = <RADAR-CONTACT-STATUS>
CALL REF (ID)
```

3.5.4 (A) The TRACK-BALL-POSITION Function

The TRACK-BALL-POSITION function, shown in Figure 36, computes the position to which the operator is to move the TRACK-BALL based upon the DESIRED and ESTIMATED values of the HOOK-POSITION, the current ESTIMATED value of the TRACK-BALL, and certain of the equipment characteristics -- specifically the GAIN factor relating movement of the TRACK-BALL to movement of the HOOK, and the current RADAR-SCALE. In order to compute the position

*Knowing the dictionary entry number enables the analyst to refer to certain data that HOS maintains on the variable that does not correspond to any of the standard parameters. These data include X, Y, and Z locations of the variable, the time it was estimated, etc.

to which the TRACK-BALL is to be moved, the DESIRED value of the HOOK-POSITION is first obtained and stored as the FORTRAN variable DHOOK by the statement:

```
DHOOK = DESIRED (<HOOK-POSITION>)
```

This positional value is then decomposed into its X and Y components and stored as the FORTRAN variables DHOOKX and DHOOKY by the statements:*

```
DHOOKX = XVALUE (DHOOK)
DHOOKY = XVALUE (DHOOK)
```

Similarly, the X and Y components of the ESTIMATED value of the HOOK-POSITION are stored as the FORTRAN variables EHOOKX and EHOOKY by the statements:

```
EHOOKX = XVALUE ('HOOK-POSITION')
EHOOKY = YVALUE ('HOOK-POSITION')
```

The new desired value for the TRACK-BALL can then be computed by taking the differences between the respective DESIRED and ESTIMATED components, multiplying by the system gain factor and adding the resultant values to the current TRACK-BALL position. These calculations are performed by the two statements:

```
XNEW = (DHOOKX-EHOOKX)*GAIN + XVALUE ('TRACK-BALL')
YNEW = (DHOOKY-EHOOKY)*GAIN + YVALUE ('TRACK-BALL')
```

*XVALUE and YVALUE are FORTRAN functions that are used in HOS to obtain the X and Y components of a positional quantity.

The X and Y components of the new TRACK-BALL position, XNEW and YNEW, are then stored as the values of TRACK-BALL-POSITION by the statement:*

```
'TRACK-BALL-POSITION' = PACKXY (XNEW,YNEW)
```

The system gain factor, GAIN, is a function of an input gain factor associated with an initial RADAR-SCALE, and the current RADAR-SCALE. The input gain factor must be divided by the current RADAR-SCALE in order to ensure that the HOOK will move the same physical distance on the screen for two equal movements of the TRACK-BALL, irrespective of the current RADAR-SCALE.

The system gain factor is obtained from input information supplied to HOS about the TRACK-BALL by the statements:

```
  IM = MODEL (<TRACK-BALL>)
GAIN = PARA (IM,9)/'RADAR-SCALE'
```

These statements reference data that is supplied to HOS at execution time -- the "model" number associated with the TRACK-BALL, and one of the system parameters (the input gain factor) associated with that model. These data will be discussed in more detail when we describe the inputs to the HOS simulation.

3.5.5 (A) Referencing Other Operator Functions

An operator function can reference another operator function using the same conventions described in Section 3.5.2 for displays, controls, and symbols. Care must be exercised, though, when a function is *recursive* -- i.e., when the function references a second function which in turn references

*The function PACKXY combines the X and Y components into a single value for storage purposes.

the first. Recursive relationships between functions *are* permitted *as long as parameters other than ESTIMATED values are being referenced.* Functions that contain recursive relationships involving ESTIMATED values (i.e., the names of the functions are in quotation marks) are *not* permitted.

There are several other restrictions relating to the referencing of one function from another. These restrictions are:

(1) An operator function cannot reference another function in such a way that the name of the second function appears in quotation marks on the left side of an equals sign. This restriction exists because the occurrence of a function name in quotation marks on the left side of an equals sign signals the end of a function definition.

(2) An operator function cannot reference itself by using the name of the function in quotation marks in the function definition -- this would be a recursive relationship. It can, however, reference itself through the use of the caret notation.

3.5.6 (A) Introductory Statements in the Operator Functions Section

The operator functions section is introduced by an OPERATOR FUNCTIONS statement. This statement is followed by any non-executable FORTRAN statements (COMMON, DIMENSION, EQUIVALENCE, or DATA statements) needed by any of the functions in the section. The first two executable statements after the non-executable statements must be:

```
      GO TO 1000
 9000 CONTINUE
```

3.5.7 (A) Other Constraints on the Operator Functions

All the rules of standard FORTRAN apply to the OPERATOR FUNCTIONS section. Several conventions should, however, be observed when coding the OPERATOR FUNCTIONS:

(1) Statement numbers between 9000 and 10000 inclusive should not be used in any OPERATOR FUNCTION.

(2) FORMAT statements should use either the H convention for Hollerith fields or the CDC 6600-specific Hollerith delimiters *...* or ≠...≠. The quotation mark Hollerith delimiter, '...', which is standard on most FORTRAN systems, should *not* be used since quotation marks are used to delimit HOPROC variables.

(3) Though it is not strictly necessary, each function should be self-contained. That is, a function should not contain a GO TO statement that transfers control to a statement in *another* function.

(4) The preferred logical unit for inputting data (in a READ statement) is logical unit 7. Logical unit 5 may also be used (with caution), but this practice is not recommended. The preferred logical unit for outputting data (via a WRITE statement) is logical unit 6. The following logical units should not be used for *any* purpose: 8,9,10,11,12,13,14. Other logical units may be referenced *if* the PROGRAM card in HOS is changed to accommodate these logical unit numbers.

(5) Although it is possible to reference the ACTUAL values of displays, controls, and symbols in operator functions, this is not a recommended practice. ACTUAL values are the precise hardware values of the displays, controls, and symbols. This information is presumably only accessible to the operator through the information absorption and recall processes. Accessing the ACTUAL values through the functions by-passes these processes.

3.6 HARDWARE PROCEDURES

The basic structure of the hardware procedures section is identical to that of the operator procedures -- the section begins with the statement HARDWARE PROCEDURES, followed by the definitions of the procedures themselves. Each procedure begins with a DEFINE statement and continues until the next DEFINE statement. A major difference between the two procedures sections, however, is in the nature of the procedures themselves. In the operator procedures section there are four special types of procedures -- ENABLE procedures, ADJUST procedures, DISABLE procedures, and MONITOR procedures. In the hardware section there is only one special type of procedure -- SIMULATE procedures.

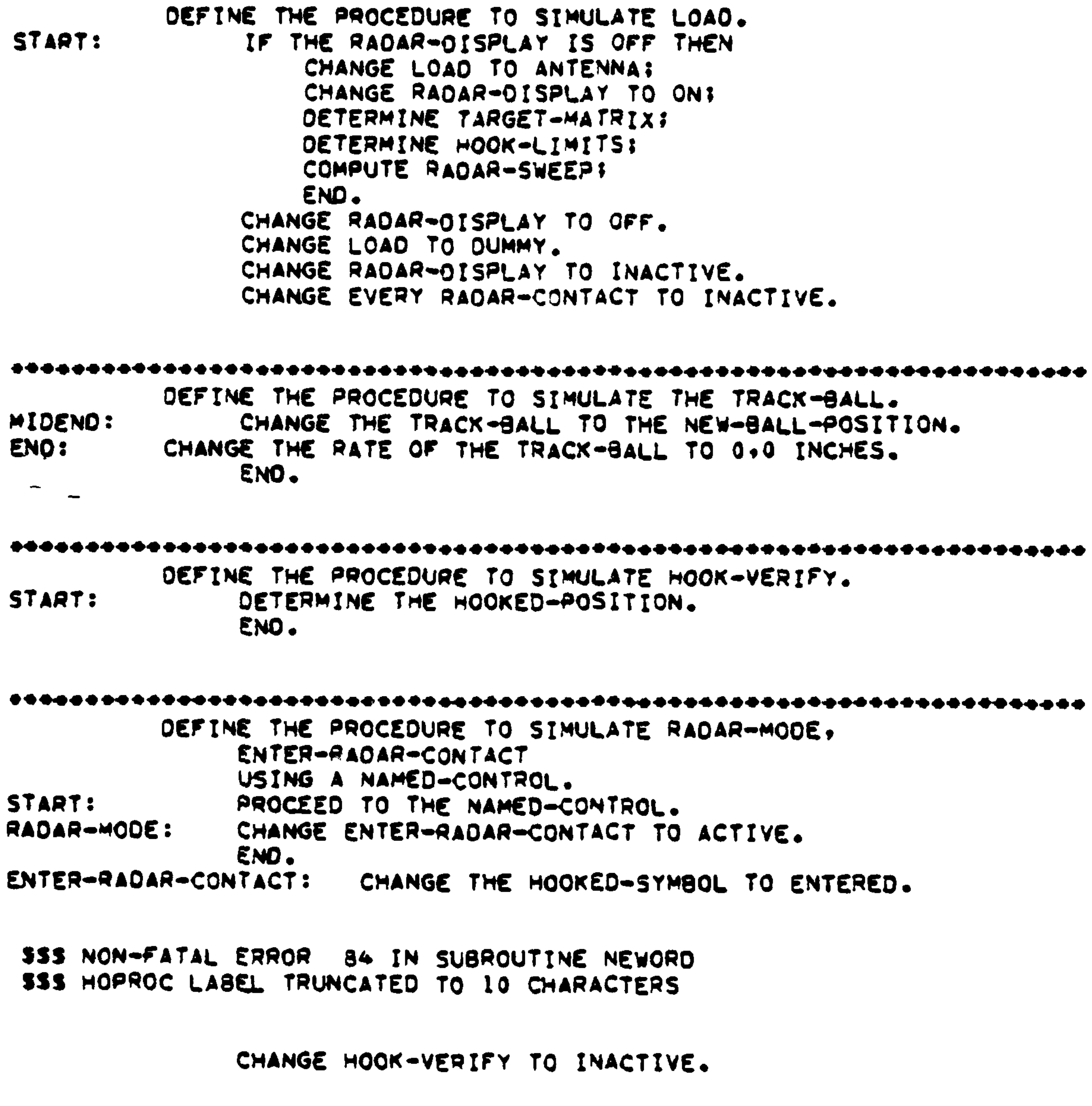
SIMULATE PROCEDURES

-- DESCRIBES THE EFFECTS OF A SPECIFIC CONTROL MANIPULATION ON OTHER DISPLAYS, CONTROLS, AND SYMBOLS.

-- CONTROLS SHOULD HAVE SIMULATE PROCEDRUES.

-- DISPLAYS AND SYNBOLS CANNOT HAVE SIMULATE PROCEDURES

-- CAN HAVE START, MIDDLE AND END ACTIONS.



```
          DEFINE THE PROCEDURE TO SIMULATE LOAD.
START:         IF THE RADAR-DISPLAY IS OFF THEN
                   CHANGE LOAD TO ANTENNA;
                   CHANGE RADAR-DISPLAY TO ON;
                   DETERMINE TARGET-MATRIX;
                   DETERMINE HOOK-LIMITS;
                   COMPUTE RADAR-SWEEP;
                   END.
               CHANGE RADAR-DISPLAY TO OFF.
               CHANGE LOAD TO DUMMY.
               CHANGE RADAR-DISPLAY TO INACTIVE.
               CHANGE EVERY RADAR-CONTACT TO INACTIVE.

************************************************************
          DEFINE THE PROCEDURE TO SIMULATE THE TRACK-BALL.
MIDEND:        CHANGE THE TRACK-BALL TO THE NEW-BALL-POSITION.
END:      CHANGE THE RATE OF THE TRACK-BALL TO 0.0 INCHES.
               END.

************************************************************
          DEFINE THE PROCEDURE TO SIMULATE HOOK-VERIFY.
START:         DETERMINE THE HOOKED-POSITION.
               END.

************************************************************
          DEFINE THE PROCEDURE TO SIMULATE RADAR-MODE,
               ENTER-RADAR-CONTACT
               USING A NAMED-CONTROL.
START:         PROCEED TO THE NAMED-CONTROL.
RADAR-MODE:    CHANGE ENTER-RADAR-CONTACT TO ACTIVE.
               END.
ENTER-RADAR-CONTACT:   CHANGE THE HOOKED-SYMBOL TO ENTERED.

 $$$ NON-FATAL ERROR  84 IN SUBROUTINE NEWORD
 $$$ HOPROC LABEL TRUNCATED TO 10 CHARACTERS

               CHANGE HOOK-VERIFY TO INACTIVE.
```

Figure 37. Hardware procedures for the radar plotting simulation.

3.6.1 SIMULATE Procedures

SIMULATE procedures, like ENABLE, ADJUST, DISABLE, and MONITOR procedures, are associated with specific devices. A SIMULATE procedure describes the effects that a specific control manipulation will have on other displays, controls, and symbols in the crewstation. For example, when the SS-3 operator turns the LOAD switch to ANTENNA, various symbols are displayed on the RADAR-DISPLAY. The procedure to SIMULATE LOAD describes what happens when the LOAD switch is changed to ANTENNA -- i.e., the RADAR-DISPLAY is turned ON and the RADAR-CONTACT symbols are activated, enabling them to be read by the operator. Every control used in the simulation *should* have an associated simulate procedure (though this is not absolutely necessary). Displays and symbols do not have simulate procedures.

Simulate procedures are structured to accommodate the fact that some control manipulations may require a considerable length of time and that some hardware activities may occur at the beginning of the manipulation, others in the middle, and still others at the end. Sections of a simulate procedure can be labeled to indicate events that occur at the beginning of the manipulation; other sections can be labeled to indicate events applicable in the middle of a manipulation, and still other events can be labeled as occurring at the end of the manipulation. When the manipulation begins, only the "start" actions will be executed. During the manipulation, only the "middle" actions will be executed, and at the end of the manipulation, only the "end" actions will be executed.

The simulate procedures for the radar plotting problem are shown in Figure 37. Three of the simulate procedures in Figure 37, SIMULATE HOOK-VERIFY, SIMULATE LOAD, and SIMULATE RADAR-MODE, use only the START label. The fourth SIMULATE procedure, SIMULATE TRACK-BALL, uses the MIDDLE* and END labels.

*The label MIDEND indicates actions to be performed *both* in the middle of a manipulation and at the end of the manipulation.

3.6.2 (A) Regular Hardware Procedures

There are also "regular" hardware procedures that are structurally identical to regular operator procedures. Any SIMULATE procedure can invoke a regular hardware procedure which, in turn, can invoke still other regular hardware procedures. Regular hardware procedures, like operator procedures, but unlike SIMULATE procedures, do not have START, MIDDLE, and END sections. Typically, regular hardware procedures are executed immediately and completely. Hardware procedures can, however, be put on an *active procedures list* similar to the active procedure list for operator procedures. The active hardware procedures list differs from the active operator procedures list in that active hardware procedures are generally executed *every time* the hardware is updated. This feature is useful for procedures that, for example, update the locations of the symbols on a display screen -- every time the hardware is updated, the locations of all the symbols on the screen will be updated to conform to their new real-world locations. The analyst can also specify a frequency with which a hardware procedure is to be executed -- for example, a hardware procedure may only have to be executed once every five seconds. Specifying an update frequency can help to cut down on the amount of computer time required for the simulation, by eliminating unnecessary hardware procedure executions.

3.6.3 The Radar Plotting Hardware Procedures

The radar plotting hardware procedures will be discussed below in the order in which these procedures are executed as determined by the actions taken by the operator. In these discussions, we will only indicate

(1) The ways in which the hardware procedures (and functions) differ from the operator procedures (and functions), and

(2) Any new constructs that have not already been discussed in the preceding sections.

3.6.4 SIMULATE Procedures for Discrete Controls

The first control action that the operator performs in the HOS radar plotting simulation is to switch the LOAD switch to the ANTENNA position. When this occurs, three hardware functions occur -- the actual value of the LOAD switch is changed to ANTENNA, the radar-display is turned on, and the locations of the targets in the real-world are displayed on the radar screen. These functions are performed by the SIMULATE LOA^ procedure.

The first statement in this procedure

```
IF THE RADAR-DISPLAY IS OFF THEN...
```

distinguishes between turning the LOAD switch from DUMMY to ANTENNA and from ANTENNA to DUMMY. When the LOAD switch is in the DUMMY position, as it will be at the beginning of the simulation, the RADAR-DISPLAY will be OFF. Changing the LOAD switch to the ANTENNA position turns the RADAR-DISPLAY ON. Therefore, if the RADAR-DISPLAY is OFF at the start of the manipulation, then the first actions to be taken are to:

```
CHANGE RADAR-DISPLAY TO ON;

CHANGE LOAD TO ANTENNA;
```

These two statements are easily recognizable as ALTER statements. However, they differ in one important respect from the ALTER statements in the operator procedures -- whereas ALTER statements in the operator procedures section implicitly referred to the DESIRED value of a device, ALTER statements in the hardware procedures section implicitly refer to the ACTUAL value of the device. Thus, these statements change the ACTUAL value of the RADAR-DISPLAY to ON and the ACTUAL value of the LOAD switch to ANTENNA.

3.6.5 Hardware Functions

The next three statements:

```
DETERMINE TARGET-MATRIX;
```

```
C
C   TARGET-MATRIX
C
      READ (7,900) NTGTS
900   FORMAT (I2)
      READ (7,901) (XT(IO),YT(IO),HT(IO),RT(IO),IO=1,NTGTS)
901   FORMAT (4F5.0)
      'TARGET-MATRIX'=0
      HOOKX=XVALUE('HOOK-POSITION')
      HOOKY=YVALUE('HOOK-POSITION')
      HOOKL='RADAR-SCALE'*'HOOK-RADIUS'/16
      UPPER(<HOOK-POSITION>)=PACKXY(HOOKX+HOOKL,HOOKY+HOOKL)
      LOWER(<HOOK-POSITION>)=PACKXY(HOOKX-HOOKL,HOOKY-HOOKL)
      'HOOK-LIMITS'=HOOKL
      DO 300 I=1,NTGTS
      XT(I) = XT(I) + RT(I)*SIN(HT(I)*PI/180.0)*(STIME-TTIME)/3600.0
      YT(I) = YT(I) + RT(I)*COS(HT(I)*PI/180.)*(STIME-TTIME)/3600.0
      CALL SYMPSTN(<RADAR-CENTER>,<RADAR-CONTACT-STATUS>+I,
     *     <RADAR-CONTACT-STATUS>+I,'RADAR-SCALE',1,XT(I),YT(I))
      STATE(<RADAR-CONTACT-STATUS>+I)=1
      STATE(<RADAR-CONTACT-POSITION>+I)=1
300   CONTINUE
      TTIME=STIME
      'RADAR-SWEEP'=1
```

Figure 38. Hardware functions for radar plotting simulation.

```
DETERMINE HOOK-LIMITS;

COMPUTE RADAR-SWEEP;
```

are easily recognizable as COMPUTE statements. These statements initiate the execution of the three hardware functions shown in Figure 38. The first function, TARGET-MATRIX, reads in a set of data cards containing the coordinates of the targets to be plotted, their speeds and headings. This function illustrates how an operator function can be used to read in data at execution time. This capability enables the HOS hardware and operator procedures to be developed independently from the "real-world data" that will obtain at the time the simulation is run. New real-world situations can then be created and run through the model without having to rewrite and procedures or recompile any program components.

The second function, HOOK-LIMITS, establishes limiting values for the HOOK-POSITION. The upper limit, UPPER (<HOOK-POSITION>) is obtained by taking the distance that converts the HOOK-RADIUS (given in inches) to the value (in miles) represented by the HOOK-RADIUS to the X and Y components of the current value of the HOOK-POSITION. The lower limit, LOWER (<HOOK-POSITION>) is obtained by subtracting the same amount from the current HOOK-POSITION. The upper and lower limits thus define a square (rather than the circle that is the actual shape of the HOOK) within which the hook must be in order to satisfy an IF... OK or IF... WITHIN LIMITS test.

The third function, RADAR-SWEEP, places the targets that were read in by the TARGET-MATRIX function onto the screen as active RADAR-CONTACTS. This function uses the subroutine SYMPSTN* to place the symbols on the screen at their current real-world locations. The statements:

```
STATE (<RADAR-CONTACT-STATUS> + I) = 1
STATE (<RADAR-CONTACT-POSITION> + I) = 1
```

*Described in more detail in the HOS Users' Guide and on the HOS Reference Card.

set the STATE of the i^{th} RADAR-CONTACT's characteristics, STATUS and POSITION, to 1 (ACTIVE), thereby enabling the operator to read these symbol characteristics.

3.6.6 (A) Altering the STATE of a Display, Control, or Symbol

If the operator had changed the LOAD switch from ANTENNA to DUMMY, the statements:

```
CHANGE RADAR-DISPLAY TO OFF.
CHANGE LOAD TO DUMMY.
CHANGE RADAR-DISPLAY TO INACTIVE.
CHANGE EVERY RADAR-CONTACT TO INACTIVE.
```

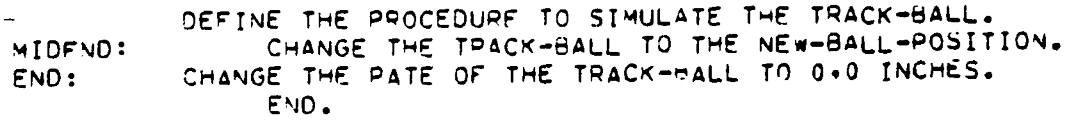
would have been executed. The first two ALTER statements are self-explanatory. The third statement changes the STATE parameter associated with the RADAR-DISPLAY to INACTIVE.* This means that the operator will once again have to execute the ENABLE procedure for RADAR-DISPLAY before he will be able to read any information from the RADAR-DISPLAY. The fourth statement changes the STATE parameter for every RADAR-CONTACT on the RADAR-DISPLAY to INACTIVE. All the characteristics of every RADAR-CONTACT, i.e., both the STATUS and POSITION characteristics, will be made INACTIVE. Consequently, the operator will not be able to read any of the RADAR-CONTACT-STATUS or RADAR-CONTACT-POSITION values without enabling the RADAR-DISPLAY.**

3.6.7 SIMULATE Procedures for Continuous Controls

The next action that the operator performs is the manipulation of the TRACK-BALL. The hardware consequences of a TRACK-BALL manipulation are described by the PROCEDURE TO SIMULATE THE TRACK-BALL (Figure 39). Since the

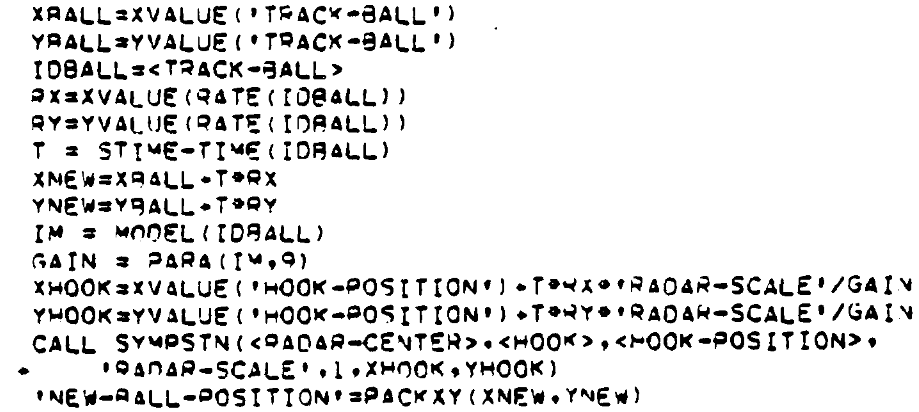
*The parameter STATE is understood implicitly because of the use of the keyword INACTIVE.

**The RADAR-CONTACT symbols have to be inactivated independently from the RADAR-DISPLAY because of the fact the HOS does not recognize that the RADAR-CONTACTs are on the RADAR-DISPLAY.



```
-              DEFINE THE PROCEDURF TO SIMULATE THE TRACK-BALL.
MIDFND:             CHANGE THE TPACK-BALL TO THE NEW-BALL-POSITION.
END:           CHANGE THE PATE OF THE TRACK-BALL TO 0.0 INCHES.
                    END.
```

Figure 39. Procedure to simulate the TRACK-BALL.



```
XBALL=XVALUE('TRACK-BALL')
YBALL=YVALUE('TRACK-BALL')
IDBALL=<TRACK-BALL>
RX=XVALUE(RATE(IDBALL))
RY=YVALUE(RATE(IDBALL))
T = STIME-TIME(IDBALL)
XNEW=XBALL+T*RX
YNEW=YBALL+T*RY
IM = MODEL(IDBALL)
GAIN = PARA(IM,9)
XHOOK=XVALUE('HOOK-POSITION')+T*RX*'RADAR-SCALE'/GAIN
YHOOK=YVALUE('HOOK-POSITION')+T*RY*'RADAR-SCALE'/GAIN
CALL SYMPSTN(<RADAR-CENTER>,<HOOK>,<HOOK-POSITION>,
+    'RADAR-SCALE',1,XHOOK,YHOOK)
'NEW-BALL-POSITION'=PACKXY(XNEW,YNEW)
```

Figure 40. The NEW-BALL-POSITION hardware function.

TRACK-BALL is a continuous (positional) device, there are hardware events that occur throughout the manipulation -- specifically the ACTUAL value of the TRACK-BALL will change throughout the manipulation and the ACTUAL value of the HOOK-POSITION will change concurrently to conform to the new trackball position. The hardware function, NEW-BALL-POSITION (Figure 40), computes the new positions for both the TRACK-BALL and the HOOK. The statement:

CHANGE THE TRACK-BALL TO THE NEW-BALL-POSITION.

sets the ACTUAL value of the TRACK-BALL to the result of the NEW-BALL-POSITION calculation (the new HOOK-POSITION is set internally within the function).

3.6.8 (A) The RATE and TIME Parameters

The NEW-BALL-POSITION function (Figure 40) references two parameters associated with the TRACK-BALL -- its RATE and its TIME. When the control manipulation is begun, HOS will automatically set the RATE of the TRACK-BALL to a value dependent on

(1) The current position of the TRACK-BALL,

(2) Where the operator will be moving the TRACK-BALL to, and

(3) The amount of time the manipulation will take.

The RATE will be assumed to be constant throughout the manipulation, after which it will be set to zero by the statement:

CHANGE THE RATE OF THE TRACK-BALL TO 0,0 INCHES.

Throughout the manipulation, the RATE parameter is available for use in calculations such as the calculation of the new HOOK-POSITION in NEW-BALL-POSITION.

```
          DEFINE THE PROCEDURE TO SIMULATE HOOK-VERIFY.
START:         DETERMINE THE HOOKED-POSITION.
               END.
          DEFINE THE PROCEDURE TO SIMULATE RADAR-MODE.
               ENTER-RADAR-CONTACT
               USING A NAMED-CONTROL.
START:         PROCEED TO THE NAMED-CONTROL.
RADAR-MODE:    CHANGE ENTER-RADAR-CONTACT TO ACTIVE.
               END.
ENTER-RADAR-CONTACT:   CHANGE THE HOOKED-SYMBOL TO ENTERED.
               CHANGE HOOK-VERIFY TO INACTIVE.
                    END.
```

Figure 41. The simulate procedures for HOOK-VERIFY, RADAR MODE and ENTER-RADAR-CONTACT.

The other parameter referenced in the NEW-BALL-POSITION calculation is the TIME parameter. This parameter represents the last time (in seconds) that the control was updated. Therefore, by subtracting this time from the current simulation time, STIME, and multiplying by the rates of movement, the change in the TRACK-BALL's position (and in the HOOK-POSITION) can be calculated.

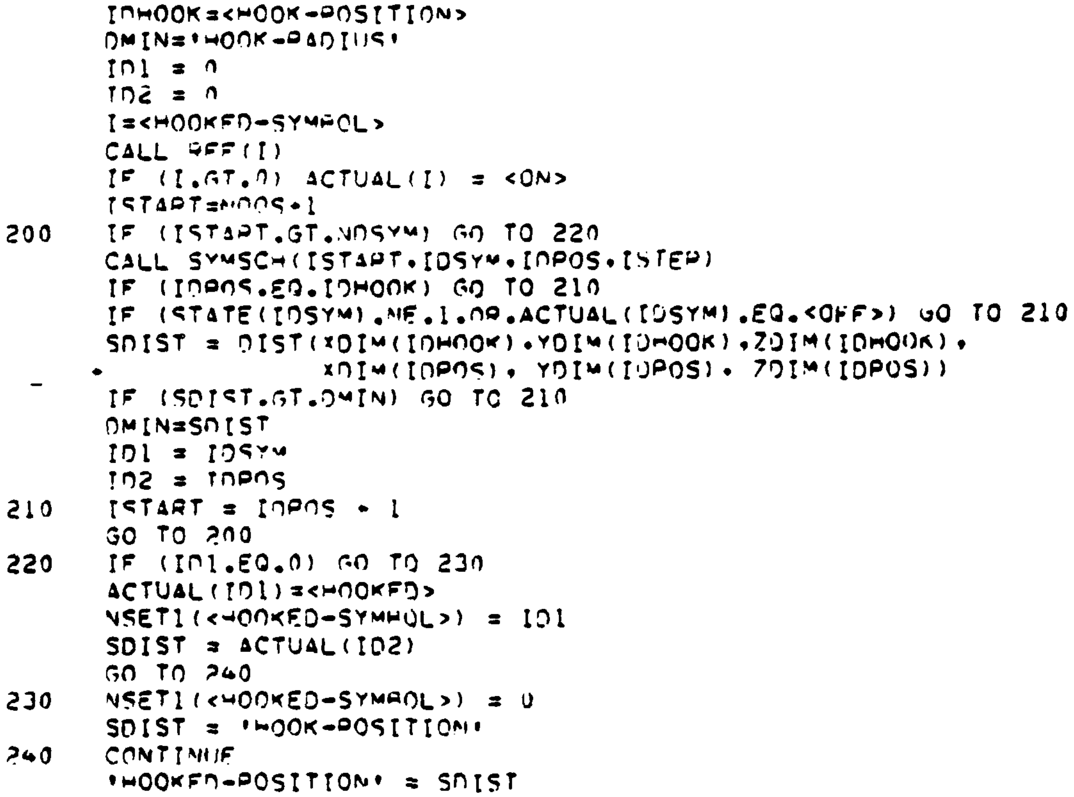
3.6.9 SIMULATE Procedures for Momentary Controls

After having manipulated the TRACK-BALL so that the HOOK is in the correct position, the operator depresses the HOOK-VERIFY pushbutton. The SIMULATE procedure that is invoked when this control manipulation is performed is shown in Figure 41 . This procedure simply calls the HOOKED-POSITION function, shown in Figure 42 , which determines which symbol has been hooked.

3.6.10 Multiple Titles in a Procedure Definition

The next two actions that the operator takes are to depress the RADAR-MODE and the ENTER-RADAR-CONTACT switches. Since both of these switches are associated with the radar-matrix functions on the keyset tray, a single SIMULATE procedure has been defined for *both* controls. This is done by simply listing the names of both controls in the DEFINE statement. As many controls as desired can be defined by a single SIMULATE procedure.* However, there is a practical limitation that results from the fact that, within the procedure, the actions that are to occur as a result of having manipulated one control as opposed to another must be distinguishable. Therefore, only controls that "belong" together and/or have several functions in common should be grouped together in a single SIMULATE procedure.

*This is also true for ENABLE, ADJUST, DISABLE, and MONITOR procedures.



```
       IDHOOK=<HOOK-POSITION>
       DMIN='HOOK-RADIUS'
       ID1 = 0
       ID2 = 0
       I=<HOOKED-SYMBOL>
       CALL OFF(I)
       IF (I.GT.0) ACTUAL(I) = <ON>
       ISTART=NDOS+1
200    IF (ISTART.GT.NDSYM) GO TO 220
       CALL SYMSCH(ISTART,IDSYM,IDPOS,ISTEP)
       IF (IDPOS.EQ.IDHOOK) GO TO 210
       IF (STATE(IDSYM).NE.1.OR.ACTUAL(IDSYM).EQ.<OFF>) GO TO 210
       SDIST = DIST(XDIM(IDHOOK),YDIM(IDHOOK),ZDIM(IDHOOK),
     +              XDIM(IDPOS), YDIM(IDPOS), ZDIM(IDPOS))
       IF (SDIST.GT.DMIN) GO TO 210
       DMIN=SDIST
       ID1 = IDSYM
       ID2 = IDPOS
210    ISTART = IDPOS + 1
       GO TO 200
220    IF (ID1.EQ.0) GO TO 230
       ACTUAL(ID1)=<HOOKED>
       NSET1(<HOOKED-SYMBOL>) = ID1
       SDIST = ACTUAL(ID2)
       GO TO 240
230    NSET1(<HOOKED-SYMBOL>) = 0
       SDIST = 'HOOK-POSITION'
240    CONTINUE
       'HOOKED-POSITION' = SDIST
```

Figure 42. The HOOKED-POSITION hardware function.

3.6.11 Arguments

The *argument* NAMED-CONTROL is used to distinguish which control is being used when the SIMULATE procedure is executed. Arguments are "dummy" HOPROC variable names that can be used to serve a variety of functions. Typically, an argument is used in place of an actual HOPROC variable in a statement in which the name of the referenced HOPROC variable is unknown at the time the HOPROC code is being written. For example, in the statement:

```
DEFINE THE PROCEDURE TO SIMULATE RADAR-MODE,

ENTER-RADAR-CONTACT USING NAMED-CONTROL.
```

the argument NAMED-CONTROL is mentioned in a *USING* clause. When either the RADAR-MODE or ENTER-RADAR-CONTACT control is used in the simulation, NAMED-CONTROL will be set so that it references the control. No matter which control is being used, the control can therefore be referenced in the SIMULATE procedure (or elsewhere) by referring to NAMED-CONTROL rather than by referring to the actual name of the control.

Some of the other ways in which arguments can be used will be described below.

3.6.12 (A) Arguments and PERFORM (START) Statements

Arguments can be used in PERFORM (or START) statements to pass values from one procedure to another. For example, the statement:

```
INCREMENT-SIMULATION-TIME USING NORMAL 180. (+-) 10.
```

will set the arguments DISTRIBUTION-TYPE, MEAN, and STANDARD-DEVIATION in the procedure INCREMENT-SIMULATION-TIME, to the values NORMAL, 189., and 10., respectively,* as a consequence of the DEFINE statement:

*The +- in parentheses is an example of a HOPROC *comment*. Words enclosed in parentheses are ignored by the HOPROC processing program HAL.

ARGUMENTS

-- DUMMY HOPROC VARIABLES

-- CAN BE USED IN:

- USING CLAUSES
- GO TO STATEMENTS
- *IN OTHER* PROCEDURAL STATEMENTS

```
DEFINE INCREMENT-SIMULATION-TIME USING DISTRIBUTION-TYPE,
MEAN, STANDARD-DEVIATION.
```

The number of values specified in the PERFORM (or START) statement must agree with the number of arguments in the DEFINE statement for the procedure being invoked by the PERFORM (or START) statement.

3.6.13 Arguments in GO TO Statements

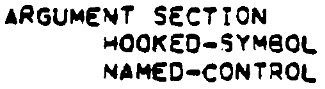
The first statement in the SIMULATE procedure for RADAR-MODE and ENTER-RADAR-CONTACT is a GO TO statement that uses the argument NAMED-CONTROL instead of a statement label. When HOS encounters this statement, it looks for a statement label that is the same as the name of the control being manipulated. Since the SIMULATE procedure defines the hardware actions for both RADAR-MODE and ENTER-RADAR-CONTACT, there must be statements in the procedure labeled with the labels RADAR-MODE and ENTER-RADAR-CONTACT. These statements identify the sections of code in the SIMULATE procedure that describe the hardware actions to occur when the respective controls are depressed.

3.6.14 Arguments in ALTER Statements

The action that is taken when ENTER-RADAR-CONTACT is depressed is to:

```
CHANGE HOOKED-SYMBOL TO ENTERED.
```

The HOPROC variable HOOKED-SYMBOL is an argument whose value is set by the hardware function HOOKED-POSITION. The argument indicates which symbol has been hooked by the operator. It will correspond, in this case, to one of the elements in the RADAR-CONTACT-STATUS subgroup.



```
ARGUMENT SECTION
        HOOKED-SYMBOL
        NAMED-CONTROL
```

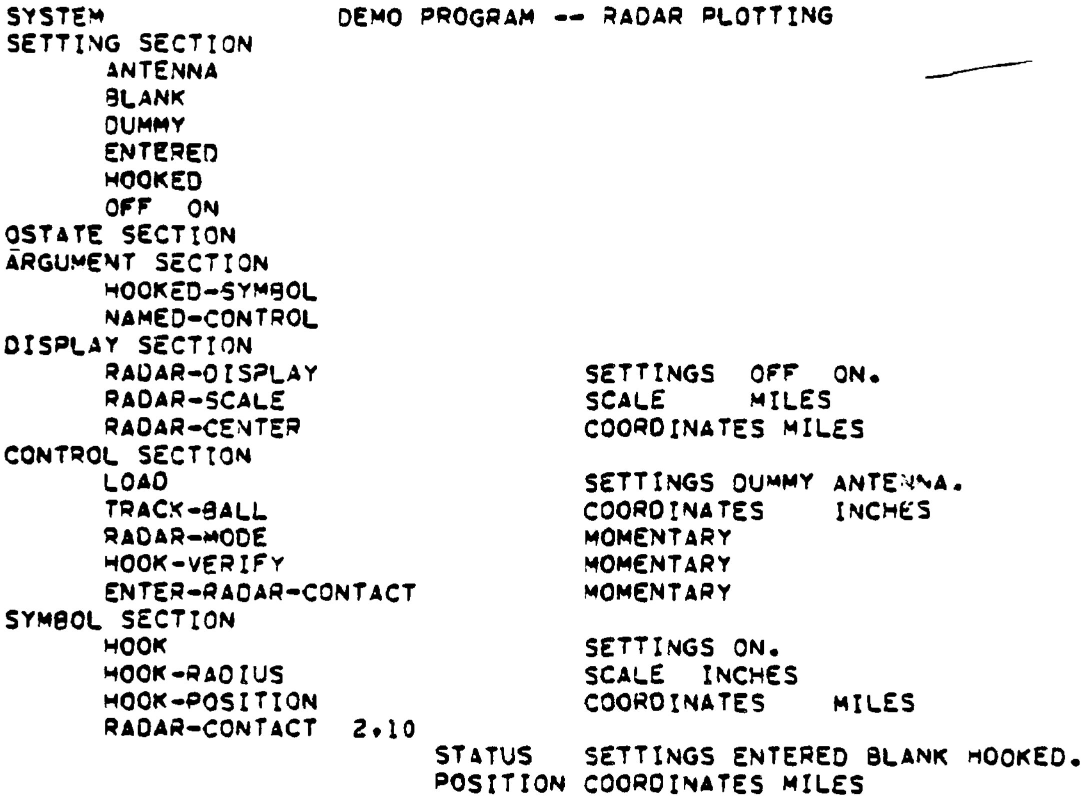
Figure 43. The ARGUMENT SECTION for the radar plotting simulation.

3.6.15 The ARGUMENT SECTION

All the argument titles used anywhere in the operator and/or hardware procedures and/or functions must be grouped together in the title declarations section. The list of arguments must be introduced by an ARGUMENT SECTION statement. Figure 43 shows the ARGUMENT SECTION used in the radar plotting simulation.

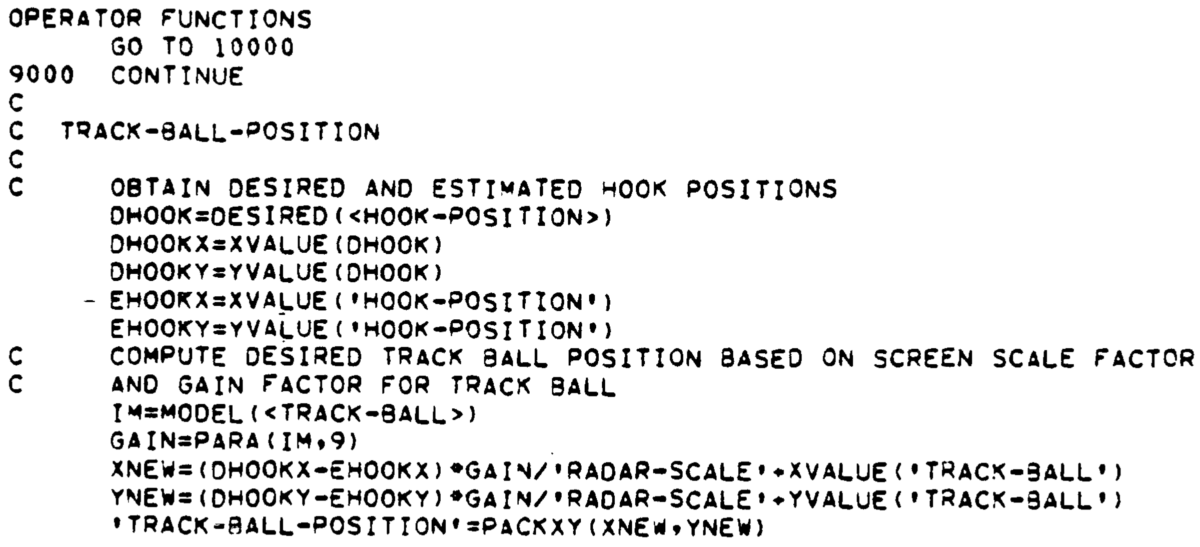
3.7 THE HOPROC DATA DECK

The complete HOPROC data deck for the radar plotting simulation is shown in Figures 44 through 48 . The deck consists of the settings, arguments, displays, controls, symbols, operator functions, hardware functions, hardware procedures, and operator procedures that have been described in the preceding sections. The first card in the data deck is an optional SYSTEM card that is used to identify the deck and to label the output that HOS will generate. The only other card(s) that may be desired in the data deck are optional cards that control certain of the outputs generated by the program (HAL) that process the HOPROC data deck. The optional output controlled by these cards, the HOS DATA DECK and the NO DATA DECK cards, will be described in Section 4.



```
SYSTEM              DEMO PROGRAM -- RADAR PLOTTING
SETTING SECTION
      ANTENNA
      BLANK
      DUMMY
      ENTERED
      HOOKED
      OFF  ON
DSTATE SECTION
ARGUMENT SECTION
      HOOKED-SYMBOL
      NAMED-CONTROL
DISPLAY SECTION
      RADAR-DISPLAY                SETTINGS  OFF  ON.
      RADAR-SCALE                  SCALE     MILES
      RADAR-CENTER                 COORDINATES MILES
CONTROL SECTION
      LOAD                         SETTINGS DUMMY ANTENNA.
      TRACK-BALL                   COORDINATES    INCHES
      RADAR-MODE                   MOMENTARY
      HOOK-VERIFY                  MOMENTARY
      ENTER-RADAR-CONTACT          MOMENTARY
SYMBOL SECTION
      HOOK                         SETTINGS ON.
      HOOK-RADIUS                  SCALE  INCHES
      HOOK-POSITION                COORDINATES    MILES
      RADAR-CONTACT  2,10
                          STATUS   SETTINGS ENTERED BLANK HOOKED.
                          POSITION COORDINATES MILES
```

Figure 44. Title declarations for the radar plotting simulation.



```
OPERATOR FUNCTIONS
      GO TO 10000
9000  CONTINUE
C
C  TRACK-BALL-POSITION
C
C     OBTAIN DESIRED AND ESTIMATED HOOK POSITIONS
      DHOOK=DESIRED(<HOOK-POSITION>)
      DHOOKX=XVALUE(DHOOK)
      DHOOKY=YVALUE(DHOOK)
      EHOOKX=XVALUE('HOOK-POSITION')
      EHOOKY=YVALUE('HOOK-POSITION')
C     COMPUTE DESIRED TRACK BALL POSITION BASED ON SCREEN SCALE FACTOR
C     AND GAIN FACTOR FOR TRACK BALL
      IM=MODEL(<TRACK-BALL>)
      GAIN=PARA(IM,9)
      XNEW=(DHOOKX-EHOOKX)*GAIN/'RADAR-SCALE'+XVALUE('TRACK-BALL')
      YNEW=(DHOOKY-EHOOKY)*GAIN/'RADAR-SCALE'+YVALUE('TRACK-BALL')
      'TRACK-BALL-POSITION'=PACKXY(XNEW,YNEW)
```

Figure 45. Operator functions for the radar plotting simulation.

```
HARDWARE FUNCTIONS
      COMMON /TARGETS/XT(10),YT(10),RT(10),HT(10)
      DATA PI/3.141592651/
      GO TO 10000
9000  CONTINUE
C
C  TARGET-MATRIX
C
      READ (7,900) NTGTS
900   FORMAT (I2)
      READ (7,901) (XT(IO),YT(IO),HT(IO),RT(IO),IO=1,NTGTS)
901   FORMAT (4F5.0)
      'TARGET-MATRIX'=0
      HOOKX=XVALUE('HOOK-POSITION')
      HOOKY=YVALUE('HOOK-POSITION')
      HOOKL='RADAR-SCALE'*'HOOK-RADIUS'/16
      UPPER(<HOOK-POSITION>)=PACKXY(HOOKX+HOOKL,HOOKY+HOOKL)
      LOWER(<HOOK-POSITION>)=PACKXY(HOOKX-HOOKL,HOOKY-HOOKL)
      'HOOK-LIMITS'=HOOKL
      DO 300 I=1,NTGTS
      XT(I) = XT(I) + RT(I)*SIN(HT(I)*PI/180.0)*(STIME-TTIME)/3600.0
      YT(I) = YT(I) + RT(I)*COS(HT(I)*PI/180.)*(STIME-TTIME)/3600.0
      CALL SYMPSTN(<RADAR-CENTER>,<RADAR-CONTACT-STATUS>+I,
     +    <RADAR-CONTACT-STATUS>+I,'RADAR-SCALE',1,XT(I),YT(I))
      STATE(<RADAR-CONTACT-STATUS>+I)=1
      STATE(<RADAR-CONTACT-POSITION>+I)=1
300   CONTINUE
      TTIME=STIME
      'RADAR-SWEEP'=1
      XBALL=XVALUE('TRACK-BALL')
      YBALL=YVALUE('TRACK-BALL')
      IDBALL=<TRACK-BALL>
      RX=XVALUE(RATE(IDBALL))
      RY=YVALUE(RATE(IDBALL))
      T = STIME-TIME(IDBALL)
      XNEW=XBALL+T*RX
      YNEW=YBALL+T*RY
      IM = MODEL(IDBALL)
      GAIN = PARA(IM,9)
      XHOOK=XVALUE('HOOK-POSITION')+T*RX*'RADAR-SCALE'/GAIN
      YHOOK=YVALUE('HOOK-POSITION')+T*RY*'RADAR-SCALE'/GAIN
      CALL SYMPSTN(<RADAR-CENTER>,<HOOK>,<HOOK-POSITION>,
     +    'RADAR-SCALE',1,XHOOK,YHOOK)
      'NEW-BALL-POSITION'=PACKXY(XNEW,YNEW)
      IDHOOK=<HOOK-POSITION>
      DMIN='HOOK-RADIUS'
      ID1 = 0
      ID2 = 0
      I=<HOOKED-SYMBOL>
      CALL REF(I)
      IF (I.GT.0) ACTUAL(I) = <ON>
      ISTART=NOOS+1
```

Figure 46. Hardware functions for the radar plotting simulation.

```
200   IF (ISTART.GT.NDSYM) GO TO 220
      CALL SYMSCH(ISTART,IDSYM,IDPOS,ISTEP)
      IF (IDPOS.EQ.IDHOOK) GO TO 210
      IF (STATE(IDSYM).NE.1.OR.ACTUAL(IDSYM).EQ.<OFF>) GO TO 210
      SDIST = DIST(XDIM(IDHOOK),YDIM(IDHOOK),ZDIM(IDHOOK),
     +              XDIM(IDPOS), YDIM(IDPOS), ZDIM(IDPOS))
      IF (SDIST.GT.DMIN) GO TO 210
      DMIN=SDIST
      ID1 = IDSYM
      ID2 = IDPOS
210   ISTART = IDPOS + 1
      GO TO 200
220   IF (ID1.EQ.0) GO TO 230
      ACTUAL(ID1)=<HOOKED>
      NSET1(<HOOKED-SYMBOL>) = ID1
      SDIST = ACTUAL(ID2)
      GO TO 240
230   NSET1(<HOOKED-SYMBOL>) = 0
      SDIST = 'HOOK-POSITION'
240   CONTINUE
      'HOOKED-POSITION' = SDIST
```

Figure 46. Hardware functions for the radar plotting simulation. (cont.)

```
HARDWARE PROCEDURES
          DEFINE THE PROCEDURE TO SIMULATE LOAD.
START:         IF THE RADAR-DISPLAY IS OFF THEN
                   CHANGE LOAD TO ANTENNA;
                   CHANGE RADAR-DISPLAY TO ON;
                   DETERMINE TARGET-MATRIX;
                   DETERMINE HOOK-LIMITS;
                   COMPUTE RADAR-SWEEP;
                   END.
               CHANGE RADAR-DISPLAY TO OFF.
               CHANGE LOAD TO DUMMY.
               CHANGE RADAR-DISPLAY TO INACTIVE.
               CHANGE EVERY RADAR-CONTACT TO INACTIVE.
          DEFINE THE PROCEDURE TO SIMULATE THE TRACK-BALL.
MIDEND:        CHANGE THE TRACK-BALL TO THE NEW-BALL-POSITION.
END:      CHANGE THE RATE OF THE TRACK-BALL TO 0,0 INCHES.
               END.
          DEFINE THE PROCEDURE TO SIMULATE HOOK-VERIFY.
START:         DETERMINE THE HOOKED-POSITION.
               END.
          DEFINE THE PROCEDURE TO SIMULATE RADAR-MODE,
               ENTER-RADAR-CONTACT
               USING A NAMED-CONTROL.
START:         PROCEED TO THE NAMED-CONTROL.
RADAR-MODE:    CHANGE ENTER-RADAR-CONTACT TO ACTIVE.
               END.
ENTER-RADAR-CONTACT:   CHANGE THE HOOKED-SYMBOL TO ENTERED.
               CHANGE HOOK-VERIFY TO INACTIVE.
                    END.
```

Figure 47. Hardware procedures for the radar plotting simulation.

```
OPERATOR PROCEDURES
          DEFINE THE MISSION.
               PERFORM RADAR-PLOT.
               END.
          DEFINE THE PROCEDURE TO RADAR-PLOT.
               ENABLE THE RADAR-DISPLAY.
               IF ANY RADAR-CONTACT-STATUS IS NOT ENTERED THEN
ENTER:              DESIGNATE IT AS THE RADAR-CONTACT OF INTEREST;
                    MOVE THE HOOK-POSITION TO THE
                      RADAR-CONTACT-POSITION;
                    DEPRESS HOOK-VERIFY;
                    DEPRESS ENTER-RADAR-CONTACT.
               IF ANOTHER RADAR-CONTACT-STATUS IS NOT ENTERED
                    THEN GO TO ENTER NOW.
               END.
          DEFINE THE PROCEDURE TO ENABLE THE RADAR-DISPLAY.
               TURN LOAD TO ANTENNA.
               END.
          DEFINE THE PROCEDURE TO ADJUST THE HOOK-POSITION.
CHECK:              READ THE HOOK-POSITION.
               IF IT IS OK THEN END.
               DETERMINE THE TRACK-BALL-POSITION.
               MOVE THE TRACK-BALL TO THE RESULT.
               IF THE RATE OF THE TRACK-BALL IS NOT 0.0 INCHES
                    THEN WAIT.
               GO TO CHECK NOW.
          DEFINE THE PROCEDURE TO ENABLE HOOK-VERIFY.
               ADJUST THE HOOK-POSITION.
               END.
          DEFINE THE PROCEDURE TO ENABLE ENTER-RADAR-CONTACT.
               DEPRESS RADAR-MODE.
```

Figure 48. Operator procedures for the radar plotting simulation.

```
job, name, core, time, PK1.

ACCOUNT, charge, password.

PACKNAM, PN = PACK006.

GET, CCEXEC/UN = CS0689.

GET, HOSEXEC/UN = CS0689.
                              (NAME = name)
CCEXEC, HOSEXEC, step 1.      {FILE = file}
                              (RUN  = run )

CCEXEC, HOSEXEC, step 2.
          :
789

data for step 1

789

data for step 2
          :

6789
```

Figure 49. Deck Structure for a HOS Simulation

4. HOW TO RUN HOS

HOS is a set of three major computer programs linked to one another by the specific problem description formulated by the analyst. The characteristics of the problem provide the data that the first HOS program, HAL, uses to generate portions of the second and third computer programs, HOS and HODAC. Consequently, there is a sequence of steps that the user must follow in order properly to get from one program to the next. At certain steps, there may be data that must be input by the analyst in order to control the simulation and the resultant output and analyses. Since the actual sequence of steps required to get from one program to another is quite complex, there is an *executive* routine that manages that transfer of data between the programs for the user. The executive routine enables any step to be executed with a *single* control card on which usually only *one* entry changes. Individual control cards can be stacked one after another and, if any errors occur at any stage in the sequence, execution will halt at that step. The user can then correct the error, rerun the step, and resume processing at the next step with a minimal amount of concern for the mechanics of interfacing with the computer.

4.1 THE DECK STRUCTURE FOR A HOS SIMULATION

The deck structure for a HOS simulations is shown in Figure 49 . The deck consists of five control cards that are always used:

(1) A *job* card to identify the job and the resources required.

(2) An *account* card to control billing charges.

(3) A PACKNAM card that instructs the computer operator to mount the private disc pack containing the HOS programs.

(4) Two GET cards that access the disc files that contain the HOS executive routines.

Table 1. Steps in a HOS Simulation.

STEP	MEMORY (in octal locations)	CPU TIME (sec)	DATA	FUNCTION
HOPROC	50000	<1	1) HOPROC code in update format	Places HOPROC code onto a data file
HAL	150000	2 - 25	1) Modifications to HOPROC in update format	Executes the HAL program
HALHOS	50000	~25	1) Crewstation data 2) Run-time data 3) FORTRAN additions	Compiles HOS, creates and prepares files for input to HOS
HOS	120000 - 200000	~20:1**	See text	Executes HOS
HOPMOD	50000	<1	1) Changes to HOPROC in update format	Modifies HOPROC code
HOSMOD	50000	<1	1) *Changes to HOS in update format* 2) LIBEDIT control cards	Facilitates correcting the simulation*
HALHOD	50000	~12	---	Compiles HODAC
HODAC	70000-200000	~20:1	1) *HODAC control cards*	Executes HODAC
UTILITIES				
LISTHAL	50000	<1		Lists HAL code
LISTHOS	50000	<1		Lists HOS code
LISTHOD	50000	<1		Lists HODAC code

* For use only by those with extensive HOS experience.

** Simulated seconds: CPU seconds

These five cards are then followed by any number of *step control cards* that invoke specific steps in the processing sequence and by the data cards for each step. The various step names are shown in Table 1, which also describes the function of each step, the required input data and the approximate time and memory requirements for each step. The steps themselves are described below in more detail.

The first card in a deck, the *job* card, assigns a name to the job and specifies the amount of core and CPU time required by the job. The time required for a job consisting of several steps is simply the sums of the times required for each step, as shown in Table 1. The core required is the maximum memory required by any individual step. Since the time and memory requirements for some of the steps can vary according to the complexity of the simulation, Table 1 shows approximate minimum and maximum values or formulas that can be used to estimate the amount of time and/or memory required. The minimum values are based on the radar plotting problem that we are using as an example. The maximum values are based on the full SS-3 simulation from which this problem has been derived.

One step control card is needed for each job step. The last step control card is followed by a ${}^{7}8_{9}$ card.* The ${}^{7}8_{9}$ card is followed by any data required by Step 1. This data is terminated by another ${}^{7}8_{9}$ card, which is then followed by any data for Step 2, etc. The last set of data is terminated by a ${}^{6}7_{8_{9}}$ card.**

*A 7, 8, and 9 punched in column 1.

**Some steps may require two or more sets of data, each of which must be separated by a ${}^{7}8_{9}$ card. Some steps may not require any data, in which case the ${}^{7}8_{9}$ card for that step must be omitted.

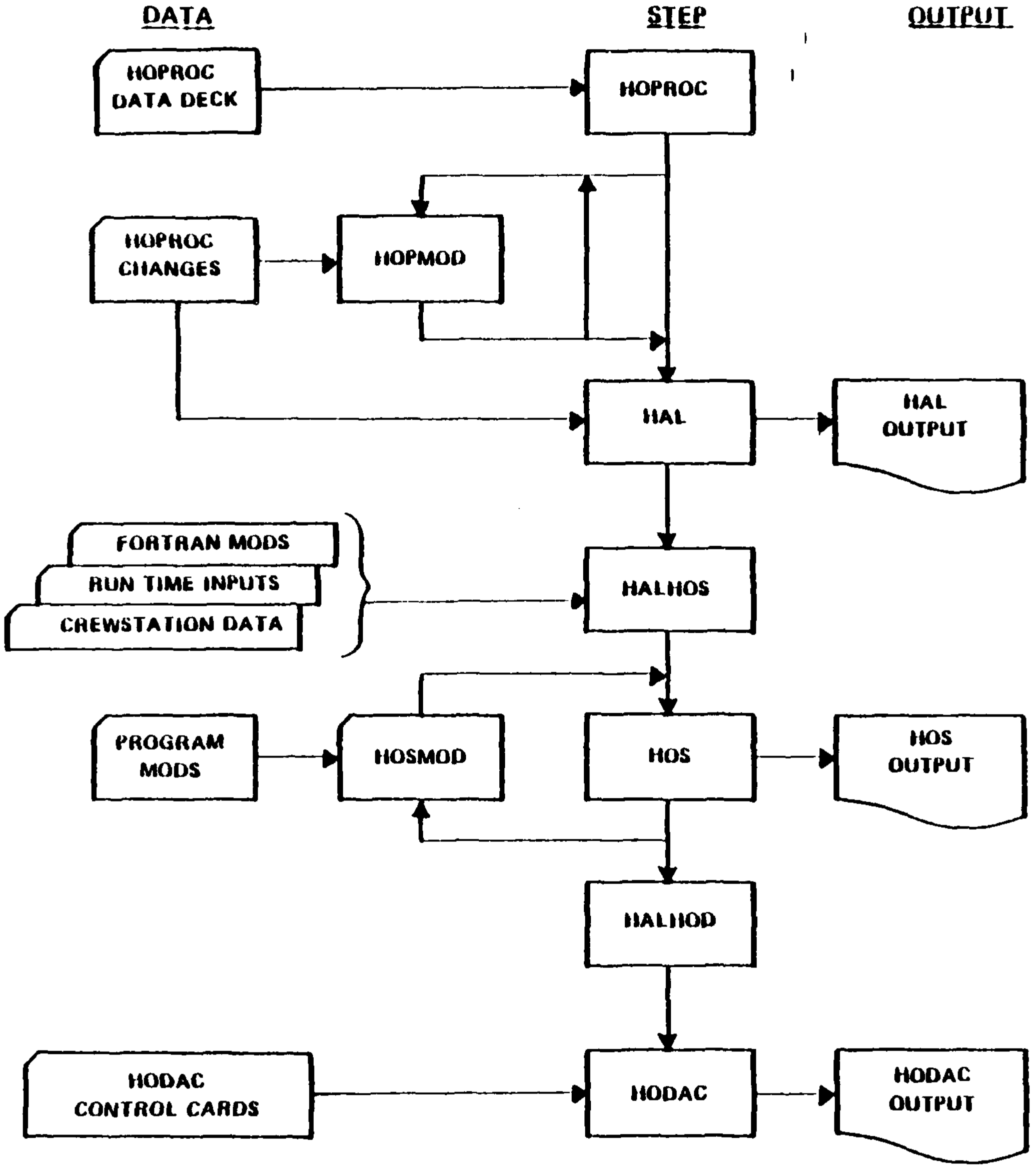


Figure 50. Flow of st in a HOS simulation.

The flow of steps to be followed in order to run a complete simulation is shown in Figure 50 . These steps are described in more detail in the following sections.

4.2 STEP HOPROC

This step reads in the HOPROC instructions describing the operator's tasks. It assigns a sequence number to each instruction so that it can be readily referenced in case corrections must be made.

The data for this step, the HOPROC instructions, can be either in the input stream or on a file already on the disc. If the data are in the input stream, then the control card supplied should be:

```
CCEXEC, HOSEXEC, HOPROC.  NAME = name, FILE = INPUT
```

where name is a one to four character identifier chosen by the analyst to identify the simulation.* If the instructions are already on a system file, then *two* control cards are necessary:

```
CCEXEC, HOSEXEC, HOPROC.  NAME = name, FILE = filename
GET, filename.
```

where *filename* is the name of the (indirect) file on which the instructions are stored.

The first statement in the HOPROC data deck must be preceded by a *DECK card. This card has the format:

```
*DECK deckname
```

*The same one to four characters must be used in all succeeding steps associated with the simulation.

where *deckname* is a name supplied by the analyst that is used by the system to label the HOPROC statements. The first card in the HOPROC data deck is an optional SYSTEM card that identifies the simulation and is used to label the pages of the output. The second card is an optional NO DATA DECK card. This card tells HAL (when it is executed) to omit the output data deck formats for HOS. These cards are followed by the title declarations sections, functions sections, and procedures sections.

4.3 STEP HAL

This step processes the code entered in step HOPROC through the HAL program. The control card required for this step is:

```
CCEXEC, HOSEXEC, HAL.  NAME = name
```

Data for this step is optional and consists of any final HOPROC changes, prepared according to the rules described in Section 4.4

4.3.1 The Output from HAL

There are four major sections in the ouput listings obtained from HAL:

(1) A listing of the HOPROC data deck and any associated error messages identified by HAL.

(2) A listing of the dictionary of HOPROC variable names and associated data.

(3) Estimates of the amount of core that will be needed in steps HOS and HODAC.

(4) A list of the data cards needed for HOS.

Of these output data, the most important are the listings of the HOPROC inputs and associated error messages. The dictionary of variable names is frequently used to explain some types of errors. The estimates of the HOS and HODAC core requirements and the listings of the HOS data cards are used to prepare the data decks for succeeding steps.

The first portion of the HAL output is a listing of all cards in the HOPROC data deck and any associated errors. Syntax errors will generally be printed on the line immediately following the line on which the error occurs. However, some types of errors may not be identified immediately by the HAL program. Such an error ususally generates an error message that does not point clearly to the source of the error. Generally, these types of errors are the result of spelling mistakes or the improper use of a keyword or a syntactic structure. Correcting the spelling mistake or the sentence structure will solve the problem.

When a serious error occurs, HAL will usually print a message stating that it is skipping to the next period, semi-colon, or AND in the deck. Consequently, any words occurring between the word that generated the error and the next period, semi-colon, or AND will not be checked for syntactical errors.

The error messages themselves are usually self-explanatory and are identified as being either informative, non-fatal, or fatal.* Although they will not prevent a program from being run through HOS, informative and non-fatal errors should be checked carefully because they generally indicate a non-standard syntactic usage which may cause the compiled program to do something other than what the analyst intended. Typically, what will happen is that an informative or non-fatal error will be symptomatic of a problem that will show up elsewhere in the simulation as a fatal error. Corrections to the data deck are made by step HOPMOD.

*A fourth class of error -- the "compiler" error -- should never be encountered by the general user. If, however, modifications are made to the current statement syntax, compiler errors might be encountered while the modifications are being debugged.

4.3.2 (A) Step HOPMOD

This step enables the user to modify the HOPROC code to correct any errors identified by HAL. The control card for this step is:

```
CCEXEC, HOSEXEC, HOPMOD.  NAME = name
```

The inputs are the corrections to the HOPROC code.

The correction set must begin with an *ID card (to identify the correction set) in the format:

```
*ID ident
```

The *ID card is followed by the set of corrections. Two basic types of corrections can be made to the HOPROC code -- deletions of lines (and the optional addition of lines of code to replace the lines in error) and insertions (to add completely new code). To delete lines of code, a *DELETE card is used:*

```
*DELETE deckname.seqno
```

where deckname.seqno is the card identifier as listed on the output from HAL. A series of consecutive cards can be deleted by entering the identifiers for the first and last cards to be deleted as follows:

```
*DELETE deckname.seqno, deckname.seqno
```

The *DELETE card can be followed by any number of lines of code to be added in place of the code being deleted.

*The abbreviation *D can be used in place of *DELETE.

Code can be added without deleting lines by means of the *INSERT card.* This card has the format:

```
*INSERT deckname.seqno
```

Any cards entered after the *INSERT card will be added *after* the card named on the *INSERT card.

4.3.3 The HOPROC Variable Dictionary

The second portion of the HAL output (Figure 51) is a listing of the dictionary of HOPROC variable names. The numeric data associated with the dictionary names can generally be ignored. It is advisable, however, to check the dictionary titles to make certain that all the variables have been correctly defined and that there are no unusual or unexpected variable names in the dictionary -- a statement sometimes gets typed in a way such that it is valid HOPROC syntax, but not what was intended. Such errors can often be spotted by scanning the dictionary.

4.3.4 HAL Estimates of HOS and HODAC Core Requirements

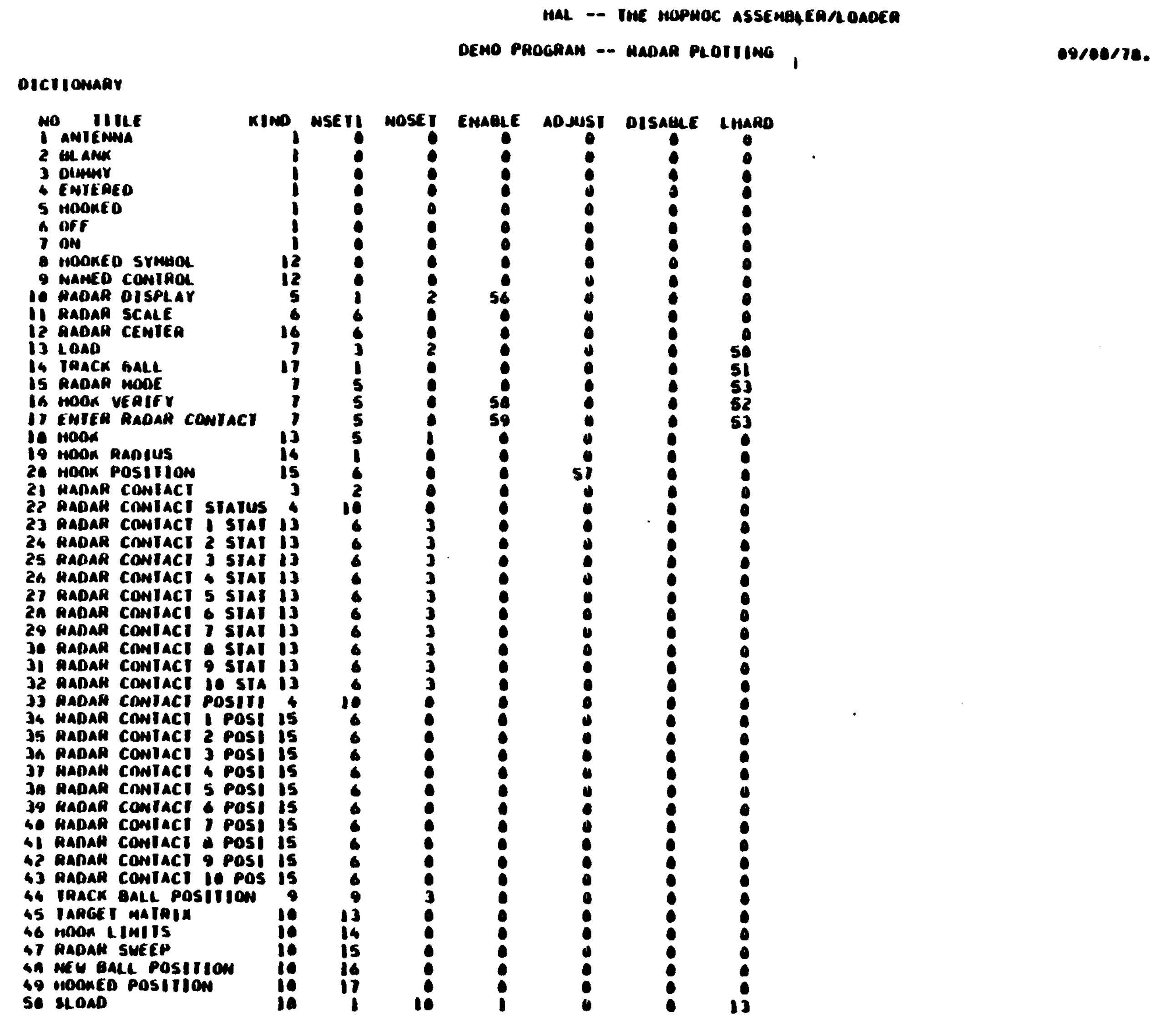
The next page of output from HAL (Figure 52) estimates the amount of core needed by steps HOS and HODAC. These estimates should be entered on the job card when these steps are run.

The HOS estimate is a rough approximation of the amount of core needed -- it should be considered to be the minimum amount that will probably be needed. The problem with estimating the HOS core requirement is that HAL has no way of telling how much machine code the hardware and operator functions will generate. If the functions are much more complicated than is usual or if they use large user-defined arrays, then HOS will require more core than the amount estimated by HAL. A simulation with a simple set of functions will require approximately the estimated amount.

*The abbreviation *I can be used in place of *INSERT.

AD-A094 353 ANALYTICS INC WILLOW GROVE PA F/G 5/8
THE HUMAN OPERATOR SIMULATOR. VOLUME IX. HOS STUDY GUIDE.(U)
SEP 78 M I STRIEB, F A GLENN, R J WHERRY N62269-78-M-6684
UNCLASSIFIED TR-1320-VOL-9 NL

3 of 4
AD A
094353



```
                                        HAL -- THE HOPHOC ASSEMBLER/LOADER

                                   DEMO PROGRAM -- RADAR PLOTTING                          09/08/78.
DICTIONARY

 NO   TITLE              KIND  NSETI  NOSET  ENABLE  ADJUST  DISABLE  LHARD
  1 ANTENNA                1     0      0      0       0       0        0
  2 BLANK                  1     0      0      0       0       0        0
  3 DUMMY                  1     0      0      0       0       0        0
  4 ENTERED                1     0      0      0       0       0        0
  5 HOOKED                 1     0      0      0       0       0        0
  6 OFF                    1     0      0      0       0       0        0
  7 ON                     1     0      0      0       0       0        0
  8 HOOKED SYMBOL         12     0      0      0       0       0        0
  9 NAMED CONTROL         12     0      0      0       0       0        0
 10 RADAR DISPLAY          5     1      2     56       0       0        0
 11 RADAR SCALE            6     6      0      0       0       0        0
 12 RADAR CENTER          16     6      0      0       0       0        0
 13 LOAD                   7     3      2      0       0       0       50
 14 TRACK BALL            17     1      0      0       0       0       51
 15 RADAR MODE             7     5      0      0       0       0       53
 16 HOOK VERIFY            7     5      0     58       0       0       52
 17 ENTER RADAR CONTACT    7     5      0     59       0       0       53
 18 HOOK                  13     5      1      0       0       0        0
 19 HOOK RADIUS           14     1      0      0       0       0        0
 20 HOOK POSITION         15     6      0      0      57       0        0
 21 RADAR CONTACT          3     2      0      0       0       0        0
 22 RADAR CONTACT STATUS   4    10      0      0       0       0        0
 23 RADAR CONTACT 1 STAT  13     6      3      0       0       0        0
 24 RADAR CONTACT 2 STAT  13     6      3      0       0       0        0
 25 RADAR CONTACT 3 STAT  13     6      3      0       0       0        0
 26 RADAR CONTACT 4 STAT  13     6      3      0       0       0        0
 27 RADAR CONTACT 5 STAT  13     6      3      0       0       0        0
 28 RADAR CONTACT 6 STAT  13     6      3      0       0       0        0
 29 RADAR CONTACT 7 STAT  13     6      3      0       0       0        0
 30 RADAR CONTACT 8 STAT  13     6      3      0       0       0        0
 31 RADAR CONTACT 9 STAT  13     6      3      0       0       0        0
 32 RADAR CONTACT 10 STA  13     6      3      0       0       0        0
 33 RADAR CONTACT POSITI   4    10      0      0       0       0        0
 34 RADAR CONTACT 1 POSI  15     6      0      0       0       0        0
 35 RADAR CONTACT 2 POSI  15     6      0      0       0       0        0
 36 RADAR CONTACT 3 POSI  15     6      0      0       0       0        0
 37 RADAR CONTACT 4 POSI  15     6      0      0       0       0        0
 38 RADAR CONTACT 5 POSI  15     6      0      0       0       0        0
 39 RADAR CONTACT 6 POSI  15     6      0      0       0       0        0
 40 RADAR CONTACT 7 POSI  15     6      0      0       0       0        0
 41 RADAR CONTACT 8 POSI  15     6      0      0       0       0        0
 42 RADAR CONTACT 9 POSI  15     6      0      0       0       0        0
 43 RADAR CONTACT 10 POS  15     6      0      0       0       0        0
 44 TRACK BALL POSITION    9     9      3      0       0       0        0
 45 TARGET MATRIX         10    13      0      0       0       0        0
 46 HOOK LIMITS           10    14      0      0       0       0        0
 47 RADAR SWEEP           10    15      0      0       0       0        0
 48 NEW BALL POSITION     10    16      0      0       0       0        0
 49 HOOKED POSITION       10    17      0      0       0       0        0
 50 SLOAD                 10     1     10      1       0       0       13
```

Figure 51. Dictionary arrays r the radar plotting simulation.

```
                                        HAL -- THE HOPKOC ASSEMBLER/LOADER

                                  DEMO PROGRAM -- RADAR PLOTTING                          09/08/78.
DICTIONARY

  NO   TITLE              KIND  NSET1  NOSET  ENABLE  ADJUST  DISABLE  LHARD
 51 $TRACK BALL             18     14     18      0       1        1     14
 52 $HOOK VERIFY            18     25     26      1       0        0     16
 53 $RADAR MODE             18     30     36      2       0        0     17
 54 MISSION                 20      1      2      0       0        0      0
 55 RADAR PLOT              20      3     11      0       0        0      0
 56 #RADAR DISPLAY          21     15     16      0       0        0      0
 57 &HOOK POSITION          22     17     24      0       0        0      0
 58 @HOOK VERIFY            21     28     29      0       0        0      0
 59 #ENTER RADAR CONTACT 21         30     30      0       0        0      0
```

Figure 51. Dictionary arrays for the radar plotting simulation. (cont.)

```
                                   HAL -- THE HOPROC ASSEMBLER/LOADER
                              DEMO PROGRAM -- RADAR PLOTTING                    09/08/78.


..................................................
ESTIMATED CORE REQUIREMENT FOR HOS EXECUTION
      CM 41400   ( DECIMAL WORDS )
      CB120670   ( OCTAL WORDS   )
..................................................


..................................................
ESTIMATED CORE REQUIREMENT FOR HODAC EXECUTION
      (ALL ANALYSES  EXCEPT DEVICES BY PROCEDURE)
      CM 28426   (DECIMAL WORDS)
      CB 067412 (OCTAL WORDS)
..................................................


..................................................
ESTIMATED CORE REQUIREMENT FOR HODAC EXECUTION
             (DEVICES BY PROCEDURE ANALYSIS)
      CM 29972   (DECIMAL WORDS)
      CB 072424 (OCTAL WORDS)
..................................................
```

Figure 52. HOS and HODAC core estimates for the radar plotting simulation.

The core estimates for HODAC are accurate, but the user should note that one of the analysis programs, the Devices By Procedure Analysis requires more core than any of the other analyses and could conceivably require more core than is available on the computer.

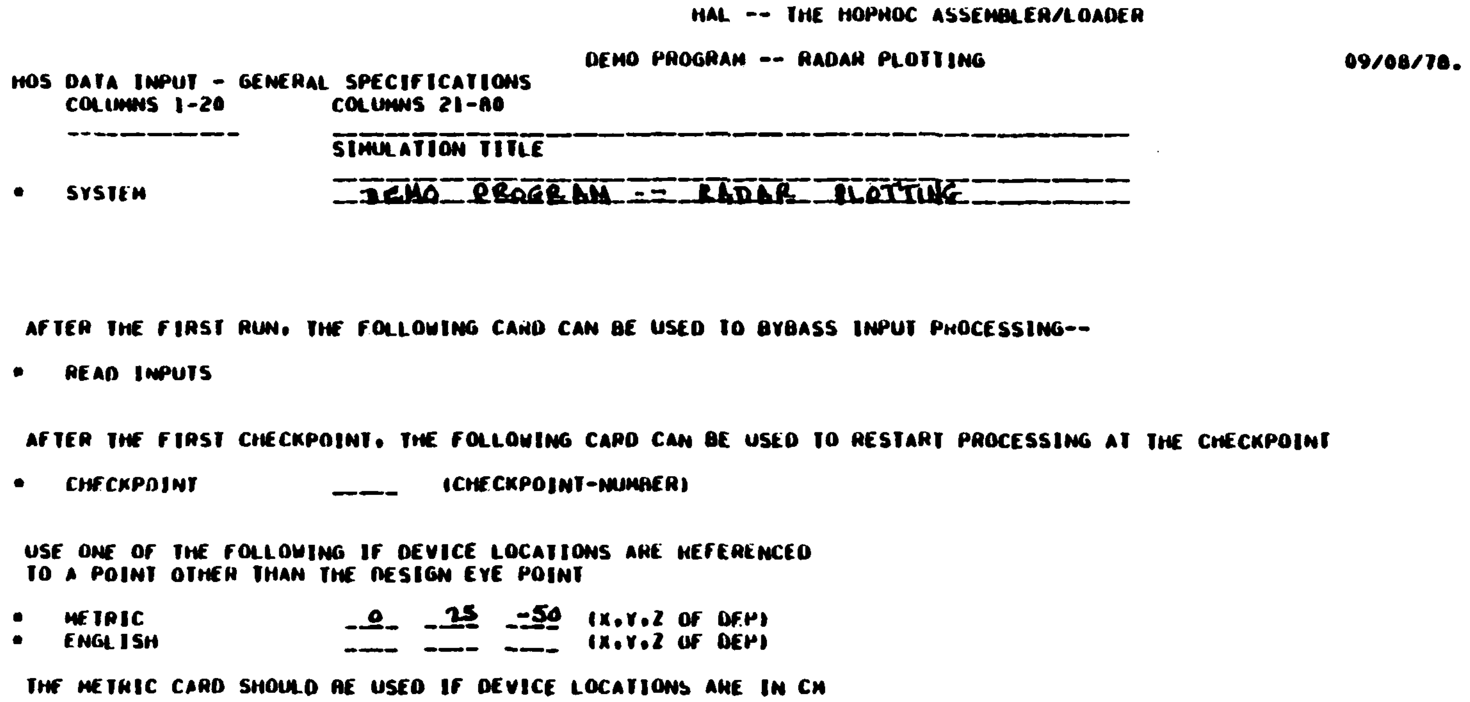
4.3.5 The HOS Crewstation Data Deck Formats

The final section of output from HAL is only printed when the HOPROC statements are error-free *or* if a HOS DATA DECK card is entered after the SYSTEM card and before the first HOPROC statement.* This output lists the data cards required to execute HOS. The format of the listings is such that the required data can be filled in and given to a keypuncher who can then punch the necessary data cards directly from the listings. Figures 52 through 59 are the listings of the HOS data deck formats for the radar plotting problem. On each page, certain lines are marked with asterisks in the left-most column. The asterisks indicate which lines are to be punched -- all the other lines simply describe the data to be entered on each card. Furthermore, some of the cards marked with an asterisk do not necessarily have to be used -- i.e., some of the cards are optional, and HOS will supply default values for these data items if the cards are omitted.

4.3.5.1 The SYSTEM Card (Figure 53)

The user can enter a title on the optional SYSTEM card that will identify the data deck and that will be used by HOS to label its output. If the SYSTEM card is not used, HOS will label the output with the title that was used on the HAL SYSTEM card (if a HAL SYSTEM card was used).

*Printing of the data deck after an error-free set of HOPROC instructions can be suppressed by entering a NO DATA DECK statement after the SYSTEM card.



```
                                                    HAL -- THE HOPHOC ASSEMBLER/LOADER

                                         DEMO PROGRAM -- RADAR PLOTTING                          09/08/78.
HOS DATA INPUT - GENERAL SPECIFICATIONS
    COLUMNS 1-20        COLUMNS 21-80
    ------------        ----------------------------------------------------------
                        SIMULATION TITLE
                        ----------------------------------------------------------
•   SYSTEM              _DEMO PROGRAM -- RADAR PLOTTING___________________________


AFTER THE FIRST RUN, THE FOLLOWING CARD CAN BE USED TO BYBASS INPUT PROCESSING--

•   READ INPUTS

AFTER THE FIRST CHECKPOINT, THE FOLLOWING CARD CAN BE USED TO RESTART PROCESSING AT THE CHECKPOINT

•   CHECKPOINT          ____    (CHECKPOINT-NUMBER)

USE ONE OF THE FOLLOWING IF DEVICE LOCATIONS ARE REFERENCED
TO A POINT OTHER THAN THE DESIGN EYE POINT

•   METRIC               _0_  _25_  -50_  (X,Y,Z OF DEP)
•   ENGLISH              ___  ___  ___  (X,Y,Z OF DEP)

THE METRIC CARD SHOULD BE USED IF DEVICE LOCATIONS ARE IN CM
```

Figure 53. HOS data deck general specifications.

```
                                   HAL -- THE HOPROC ASSEMBLER/LOADER

                                DEMO PROGRAM -- RADAR PLOTTING                        07/24/78.
HOS DATA INPUT - DISPLAY SECTION
    COLUMNS 1-20        COLUMNS 21-80
    ------------        ------------------------------------------------------------
                        MODFL      INITIAL     INITIAL    INITIAL                  COORDINATES              INITIAL
                                   MAB
                        NO.        STRENGTH    STATE      CRITICALITY    X         Y          Z             VALUE
                        -------    --------    -------    -----------   -------   -------    -------       --------
 *  DISPLAY SECTION
 *  RADAR DISPLAY       __25___    ___0____    ___1___    ___0.5_____   ___0___   __26___    __28___       __OFF___
 *  RADAR SCALE         __14___    ___0____    ___1___    ___0.5_____   -10.23_   _30.3__    _43.25_       ___32___
 *  RADAR CENTER        __25___    ___0____    ___1___    ___0.5_____   ___0___   __26___    __28___       __0 0___
```

Figure 54. HOS data deck display section.

HAL -- THE HOPHOC ASSEMBLER/LOADER

DEMO PROGRAM -- RADAR PLOTTING 07/24/78.

HOS DATA INPUT - CONTROL SECTION

COLUMNS 1-20	COLUMNS 21-80							
	MODEL	INITIAL HAB	INITIAL	INITIAL		COORDINATES		INITIAL
	NO.	STRENGTH	STATE	CRITICALITY	X	Y	Z	VALUE
• CONTROL SFCTION								
• LOAD	1	0	1	.5	-21.2	18.8	66	DUMMY
• RADAR MODE	23	0	0	0.5	-6.5	12.15	1.8	0
• HOOK VERIFY	23	0	0	0.5	-10.6	0	0	0
• ENTER RADAR CONTACT	23	0	0	0.5	-9	16.25	3.6	0
• TRACK BALL	20	0	1	0.5	0	0	0	0 0

Figure 55. HOS data deck control section.

HAL -- THE HOPROC ASSEMBLER/LOADER

DEMO PROGRAM -- RADAR PLOTTING 07/24/78.

HOS DATA INPUT - SYMBOL SECTION

COLUMNS 1-20	COLUMNS 21-80							
	MODEL	INITIAL HAB	INITIAL	INITIAL		COORDINATES		INITIAL
	NO.	STRENGTH	STATE	CRITICALITY	X	Y	Z	VALUE
• SYMBOL SECTION								
• RADAR CONTACT								
• RADAR CONTACT STATUS	16	0	0	0.5	0	26	28	BLANK
• RADAR CONTACT 1 STAT								
• RADAR CONTACT 2 STAT								
• RADAR CONTACT 3 STAT								
• RADAR CONTACT 4 STAT								
• RADAR CONTACT 5 STAT								
• RADAR CONTACT 6 STAT								
• RADAR CONTACT 7 STAT								
• RADAR CONTACT 8 STAT								
• RADAR CONTACT 9 STAT								
• RADAR CONTACT 10 STA								
• RADAR CONTACT POSITI	18	0	0	0.5	0	26	28	0 0
• RADAR CONTACT 1 POSI								
• RADAR CONTACT 2 POSI								
• RADAR CONTACT 3 POSI								
• RADAR CONTACT 4 POSI								
• RADAR CONTACT 5 POSI								
• RADAR CONTACT 6 POSI								
• RADAR CONTACT 7 POSI								
• RADAR CONTACT 8 POSI								
• RADAR CONTACT 9 POSI								
• RADAR CONTACT 10 POS								
• HOOK	16	0	1	0.5	0	26	28	ON
• HOOK RADIUS	17	0	1	0.5	0	26	28	.125
• HOOK POSITION	18	0	1	0.5	0	26	28	0 0

Figure 56. HOS data deck symbol section.

```
                                                    HAL -- THE HOPROC ASSEMBLER/LOADER

                                             DEMO PROGRAM -- RADAR PLOTTING                              09/08/78.
HOS DATA INPUT - OPERATOR FUNCTIONS
    COLUMNS 1-20        COLUMNS 21-80
    ------------        ------------------------------------------------------------------
                        FUNCTION    INITIAL      INITIAL     INITIAL       COMPUTATION   INITIAL
                                    HAB
                        TYPE        STRENGTH     STATE       CRITICALITY   TIME          VALUE
                        --------    ---------    --------    ----------    ----------    ---------
*   OPERATOR FUNCTIONS
*   TRACK BALL POSITION --------    ---------    --------    ----------    ----------    ---------
```

Figure 57. HOS data deck functions section.

HAL -- THE HOPROC ASSEMBLER/LOADER

DEMO PROGRAM -- RADAR PLOTTING 07/24/78.

HOS DATA INPUT - MODEL SPECIFICATIONS
COLUMNS 1-20 COLUMNS 21-80

MODEL TITLE	NO. OF PARAMETERS	BODY PART REQD TO ABSORB/MAN	ABSORBTION/ MANIPULATION ACCURACY	MICRO-ABSORB TIME CHARGE	DEVICE WIDTH	MODEL TYPE	ENTRY-1	ENTRY-2	FORCE REQD TO TURN
MODFL SPECIFICATIONS									
1. ROTARY SWITCH A	7	2	0	0.04	3.0	3	0	.36	
2. ROTARY SWITCH B	7	2	0	0.04	2.2	3	0	.36	
3. ROTARY SWITCH C	7	2	0	0.04	1.8	3	0	.36	
4. ROTARY SWITCH D	7	2	0	0.04	1.6	3	0	.36	
5. ROTARY SWITCH E	7	2	0	0.04	1.5	3	0	.36	
6. ROTARY SWITCH F	7	2	0	0.04	1.4	3	0	.36	
7. ROTARY SWITCH G	7	2	0	0.04	1.2	3	0	.36	
8. TOGGLE SWITCH	7	2	0	0.04	0.6	3	0	.26	
9. VARIAC A	8	2	0	0.05	2.2	2	360	360	110
10. VARIAC B	8	2	0	0.05	2.7	2	360	360	110
11. VARIAC C	8	2	0	0.05	1.6	2	360	360	110
12. IDIOT LIGHT	5	1	0	0.04	0	3			
13. INDICATOR LIGHT	5	1	0	0.04	0	3			
14. NUMERIC DISPLAY	5	1	0	0.04	0	3			
15. CONTINUOUS SYMB	5	1	.02	0.04	0	3			
16. DISCRETE SYMBOL	5	1	0	0.04	0	3			
17. CONTINUOUS SYMB	5	1	.02	0.04	0	2			
18. POSITIONAL SYMB	5	1	.02	0.04	0	2			
19. BAND MIKE	7	2	0	0.04	5.1	3	0	.5	
20. TRACKBALL	7	3	.02	0.04	4.4	2	360	360	71 10
21. FOOTSWITCH	7	4	0	0.04	7.2	3	0	3.5	
22. PUSHBUTTON A	7	2	0	0.04	2.2	3	0	.5	
23. PUSHBUTTON B	7	3	0	0.04	2.2	3	0	.5	
24. THUMBWHEEL	7	2	0	0.04	0.3	3	1	1	
25. SCREEN DISPLAY	5	1	.02	0.04	0	2			
26. PUSH TO TURN A	8	2	0	0.04	1.8	2	360	360	170
27. VARIAC D	8	2	0	0.05	1.8	2	360	360	110
28. PUSH TO TURN B	8	2	0	0.04	1	2	360	360	170
29. VARIAC E	8	2	0	0.05	1	2	360	360	170
30. FLIR GRIP SWITCH	7	3	0	0.04	2.5	3	0	.26	
END OF MODEL SPEX									

```
DISCRETE DEVICES:    ENTRY-1 = FULL ROTATIONAL CAPABILITY               BODY PARTS:  EYES        = 1
                     ENTRY-2 = TIME TO MANIPULATE THRU 1 SETTING                     EYES/HANDS  = 2
CONTINUOUS DEVICES:  ENTRY-1 = LEFT MOST TURN                                        HANDS       = 3
                     ENTRY-2 = RIGHT MOST TURN                                       FEET        = 4
```

Figure 58. HOS data deck model specifications.

```
                                         HAL -- THE HOPROC ASSEMBLER/LOADER

                                   DEMO PROGRAM -- RADAR PLOTTING                          07/24/78.
HOS DATA INPUT - HUMAN OPERATOR SPECIFICATIONS
    COLUMNS 1-20        COLUMNS 21-80
    ------------        ------------------------------------------------------------
                                                      EYE          EXTENDED     EXTENDED
                        INITIAL FIXATION COORDINATES  FIXATION     ARM          LEG
                           X         Y          Z     TOLERANCE    LENGTH       LENGTH          DEFAULT VALUES
                        -------   -------   -------   ---------    --------     --------
*   HUMAN OPERATOR SPEX
*   EYES                ___0___   ___51__   __-22__   ____0____    ___75___     ___92___   ( 12,  0,  0)(.01, 35, 48)



    COLUMNS 1-20        COLUMNS 21-80
    ------------        ------------------------------------------------------------
                                  INITIAL (AND RELAXED) BODY PART LOCATIONS
                                  RIGHT                                   LEFT
                           X         Y          Z          X          Y          Z
                        -------   -------   -------    -------    -------    -------
*   HANDS               __10.5__  ___25__   __-50__    __-10.5_   ___25__    __-50__   ( 6,  6,-18)( -6,  6,-18)
*   FEET                ___32__   ___55__   __122__    __-32__    ___55__    __-122_   ( 6, 18,-47)( -6, 18,-47)
*   SHOULDERS           ___21__   ___0___   __-16__    __-21__    ___0___    __-16__   ( 6,  0, -6)( -6,  0, -6)
*   HIPS                ___17__   ___0___   __-75__    __-17__    ___0___    __-75__   ( 6,  0,-28)( -6,  0,-28)
*   END OF HUMAN SPEX
```

Figure 59. HOS data deck operator specifications.

```
                                                 HAL -- THE HOPROC ASSEMBLER/LOADER

                                           DEMO PROGRAM -- RADAR PLOTTING                            09/08/78.
HOS DATA INPUT - OPTIONAL SPECIFICATIONS
    COLUMNS 1-20         COLUMNS 21-80
    ----------          -------------------------------------------------
                         REMEM  SURE  SMALL EXFAC  THRESH HABFAC HABTOL TINC  TDIF  TRELAX HABCY  REMCY  HABTIM REMTIM
                         ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
 *  RUN PARAMETERS       ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
    (DEFAULT VALUES)       .12    .7   .01    2.    .2    1.   .01    5.    5.    5.   10.   10.    .5    .5
 *  PRINT MESSAGES       -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 *  SUPPRESS MESSAGES    -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 *  ACTIVE MILESTONES    -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 *  INACTIVE MILESTONES  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 *  TIMED MILESTONES     -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 *  TIMED SNAPSHOTS      -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 *  TIMED ENDPOINT       --
 *  DONT PLOT ASTERISKS  ..
 *  PLOT ASTERISKS          .. ASTERISKS IN COLUMN 80 ON DISPLAY/CONTROL/SYMBOL CARDS WILL GENERATE GRAPH-TAPE OUTPUT
 *  PLOT ALL DATA
```

Figure 60. HOS data deck optional data.

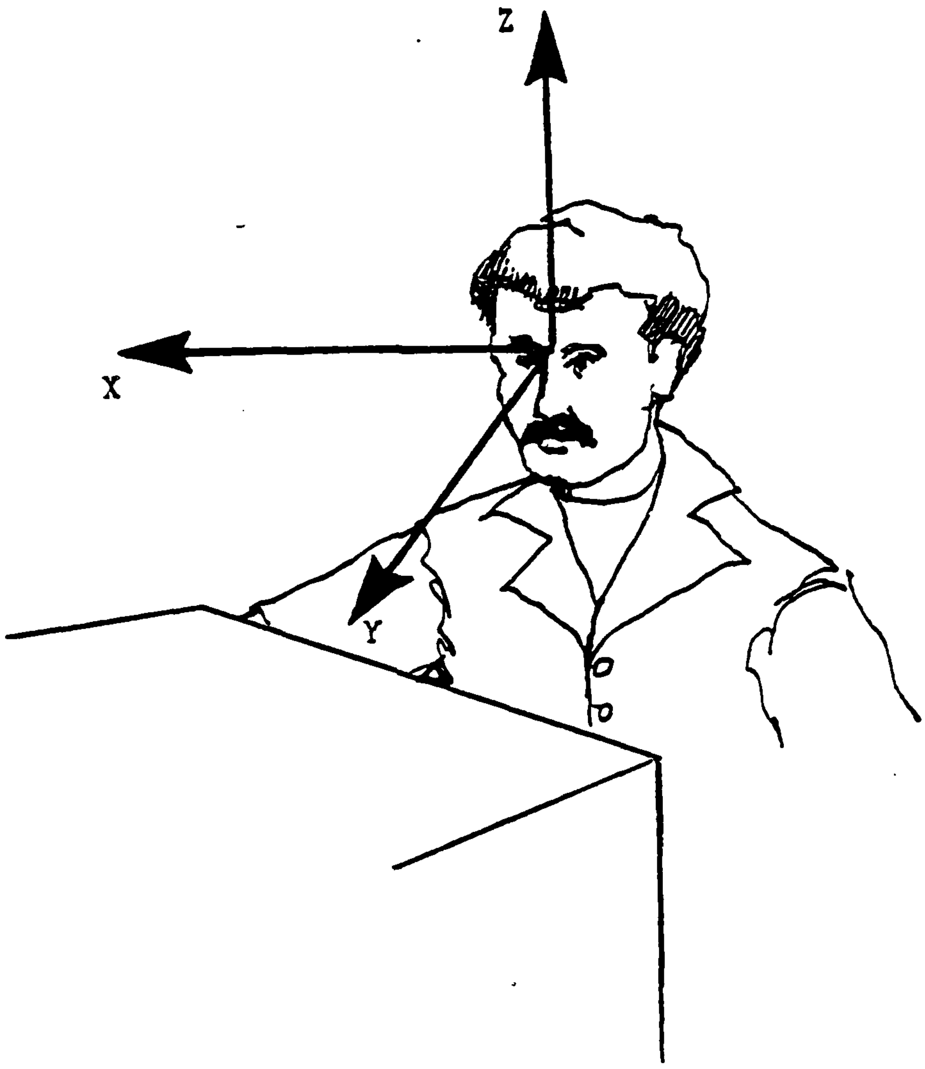


Figure 61. The design eye reference coordinate system.

4.3.5.2 (A) The READ INPUTS Card (Figure 53)

Processing the HOS data deck is a potentially time consuming process. Therefore, as the data deck for a particular problem is being processed, HOS outputs the data to a file. If the same problem is rerun with only minor modifications, HOS can obtain the bulk of the inputs from the data file that it created on the earlier run, thereby saving a significant amount of computer time. When this option is used, the analyst need only enter those inputs that are different from the earlier run. The READ INPUTS card tells HOS to use the data from the file of processed inputs and to expect only changes to that file.

4.3.5.3 (A) The CHECKPOINT Card (Figure 53)

The MILESTONE statement generates a CHECKPOINT file that contains all the data HOS needs to resume processing at the MILESTONE. The CHECKPOINT card identifies the checkpoint file number (printed in the HOS listings at the time the MILESTONE instruction is executed) at which processing is to resume. Crewstation data cards can be entered after the CHECKPOINT card to modify any of the crewstation data.

4.3.5.4 (A) The METRIC/ENGLISH Card (Figure 53)

Data on display, control, and symbol locations are stored internally in HOS in inches, referenced to a coordinate system centered at the Design Eye Point and oriented as shown in Figure 61.

Since data from blueprints or other sources may reference the locations of displays, controls, and symbols to a point other than the Design Eye Point and may use metric measurements rather than English units, HOS allows the user to specify an alternate point from which the device locations are to be measured. This alternate reference point is entered on the optional METRIC/ENGLISH card, which also indicates whether a metric to English conversion is needed.

For example, the card

```
ENGLISH         19.6     20.2     35.0
```

indicates that the display, control, and symbol coordinates are in inches and are measured from a point whose X, Y, and Z coordinates are displaced 19.6, 20.2, and 35.0 inches from the Design Eye Point. The card

```
METRIC          25       50       0
```

says that the coordinates are in centimeters, measured from a point whose X, Y, and Z coordinates are displaced 25, 50, and 0 cm, respectively, from the Design Eye Point.

4.3.5.5 The Display, Control, and Symbol Sections

These display, control, and symbol sections of the HOS crewstation data deck (Figures 54 through 56) are used to enter data on the displays, controls, and symbols in the operator's crewstation. The most important items of information entered in these sections are the *model number* of the display, control, or symbol and its *X, Y, and Z coordinates*.

The model number refers to the entry in the *Model Specifications Section* (see Section 4.3.5.7) that corresponds to the specific piece of equipment being described. Assume, for example, that several controls represent a specific type of toggle switch. That type of toggle switch would be assigned a model number and would be described in the model specifications section. All controls that are that type of toggle switch would be assigned the same model number.

The X, Y, and Z coordinates must be given with respect to the units and reference point identified on the METRIC/ENGLISH card. If no METRIC/ENGLISH card was used, the X, Y, and Z coordinates must be in inches from the Design Eye Point with the axes oriented as shown in Figure 61.

The other parameters on the display, control, and symbol section cards are the initial hab strength (usually zero), the initial state (either zero for inactive or one for active), the initial criticality (a value between zero and one), and the initial value (or setting) of the display, control, or symbol. Advice on the establishment of the hab strength and criticality parameter values is presented in the Appendix.

Generally, one data card must be entered for every display, control, symbol or symbol characteristic in the crewstation. The exception to this rule is that a single data card may be used for all the elements in a group or subgroup if every element in the group or subgroup has the same set of initial values for symbols and symbol characteristics.

4.3.5.6 The Operator Functions Section (Figure 56)

Data for the operator functions is comparable to that for the displays, controls, and symbols with the exception that no X, Y, and Z locations are specified. In addition, instead of a model number, a *function type* must be specified. The function type indicates whether the value of the function is real or integer and whether the function value is extrapolatable or not.

4.3.5.7 The Model Specifications Section

The primary function of the Model Specifications Section is to provide a *generic* listing of the types of displays, controls, and symbols used in the crewstation and their operating characteristics. A sample Model Specifications Section is shown in Figure 57. As more experimentation is done with HOS and with specific display/control/symbol characteristics, we expect that we will ultimately be able to develop a "standard" set of model specifications that will contain the data needed to model any standard display, control, or symbol accurately.

The data on the Model Specification data cards include:

- The body part (eyes, eyes/hands, hands, or feet) needed to absorb and/or manipulate the device.
- The accuracy with which the device can be absorbed and/or manipulated (i.e., the percent tolerance on absorptions and manipulations).
- The micro-absorption time charge.
- The size of the device.
- The model *type* (real/integer, extrapolatable/not-extrapolatable).
- Control specific data -- e.g., rotation limits, forces, etc.

4.3.5.8 Human Operator Specifications (Figure 58)

The next section of the data deck defines certain of the characteristics of the HOS operator -- the initial (relaxed) locations of the operator's eyes, hands, feet, shoulders, and hips, the lengths of the operator's arms and legs, and the maximum distance from the operator's eye fixation point at which an object will still be assumed to be "in focus."

4.3.5.9 Run Parameters (Figure 59)

This optional card assigns data values to some of the parameters used within the HOS program.

4.3.5.10 The PRINT/SUPPRESS MESSAGES Cards (Figure 59)

These cards control how "verbose" the output from HOS will be. Either card can be used to control which messages will be printed by the program and which will be omitted.

4.3.5.11 The ACTIVE/INACTIVE MILESTONES Cards (Figure 59)

These cards specify which MILESTONE instructions will generate listings of the dictionary arrays in HOS.

4.3.5.12 The TIMED MILESTONES Card (Figure 59)

This card can be used to generate printouts of the HOS dictionary arrays at specific times during the simulation.

4.3.5.13 The TIMED SNAPSHOTS Card (Figure 59)

This card is used to suppress simulation outputs for certain time periods and to print them for other periods.

4.3.5.14 The TIMED ENDPOINT Card (Figure 59)

This card will automatically terminate the simulation at the specified simulation time.

4.3.5.15 The PLOT Cards (Figure 59)

The DON'T PLOT ASTERISKS, PLOT ASTERISKS, and PLOT ALL DATA cards are used to control the data sent by HOS to the HODAC plotting routines.

4.4 STEP HALHOS

Step HALHOS is run after the HOPROC statements have been successfully run through HAL. It is at this step that the crewstation data described in the preceding sections is entered. In addition, data that will be needed by HOS at execution time* can also be entered at this time as can any additional FORTRAN code not already in the program (e.g., user-supplied subroutines, etc.). The control card needed to execute this step is:

```
CCEXEC, HOSEXEC, HALHOS.   NAME = name
```

The first set of input data for this step must be the crewstation data. If there is any run-time data to be supplied, this must be entered after the $^{7}8_{9}$ card that terminates the crewstation data. *If there is no run-time data, there must be an extra $^{7}8_{9}$ card in the data deck, unless this step is the last step in the run or unless any succeeding steps do not require any input data,* in which case the crewstation data will be terminated by a $^{6}7_{8_{9}}$ card.

Following the run-time data (or the $^{7}8_{9}$ card that holds its place), the analyst can enter any additional FORTRAN code modifications that may be needed for a simulation. *If there are no FORTRAN code modifications, there must be an extra $^{7}8_{9}$ card in the data deck, unless this step is the last step in the run or unless any succeeding steps do not require any input data,* in which case the modifications will be terminated by a $^{6}7_{8_{9}}$ card.

*Hardware and operator functions *may* need data at execution time. This data is read by the functions from logical unit 7.

4.5 (A) STEP HOSMOD

Any FORTRAN errors in the hardware or operator functions will be identified by step HALHOS. These errors can be corrected either by re-running the HOPROC code through HAL (using either HOPMOD to correct the HOPROC code or by correcting the original HOPROC deck and starting all over again) or by correcting the FORTRAN directly using step HOSMOD. *Correcting anything more than simple FORTRAN errors in this way* (i.e., making changes directly to the HOS code or the data generated by HAL) *is recommended only for those having advanced experience with HOS and with FORTRAN programming.* The control card needed to run step HOSMOD is:

```
CCEXEC, HOSEXEC, HOSMOD.   NAME = name
```

The data to be entered in this step are the deletions and insertions to the HOS FORTRAN code or to the data generated by HOS. This correction set follows the same general rules described for the correction set used in step HOPMOD. If step HALHOS has just been run and has identified any FORTRAN errors in the hardware or operator functions, the following data cards are required after the $^{7}8_{9}$ card that terminates the set of corrections:

```
*D COS/HFUNC                 } if there were FORTRAN errors
*I REL/HARSIM, REL/HFUNC     } in the hardware functions (HFUNC)

*D COS/MFUNC                 } if there were FORTRAN errors
*D REL/KIND, REL/MFUNC       } in the operator functions (MFUNC)
```

These cards are needed only until both HFUNC and MFUNC have been successfully compiled *once*. However, even after HFUNC and MFUNC have been successfully compiled, an extra $^{7}8_{9}$ card will be needed after the correction set, if there are succeeding steps that have inputs.

Warning: If the first set of corrections to the FORTRAN code fails to correct all the errors, then none of the changes entered will have been made permanent -- i.e., the same changes will have to be entered again along with any additional changes needed to make the subroutines compile successfully.

4.6 STEP HOS

This step executes the HOS program. The following control card is used:

```
CCEXEC, HOSEXEC, HOS.    NAME = name
```

Normally, no additional inputs are required.

4.6.1 The Output from HOS

Figures 62 through 64 are examples of the outputs from HOS for the radar plotting simulation. There are three major sections to the HOS outputs:

(1) Listings of the input arrays passed from HAL to HOS (Figure 62).

(2) Listings of the crewstation input data and any associated error messages (Figure 63).

(3) The simulation results (Figure 64).

The data arrays (Figure 62) passed from HAL to HOS will generally be of little concern to the novice HOS user. Those with more HOS experience can use these data to make simple corrections to the HOPROC instructions, thereby avoiding the need to rerun HAL.

The crewstation data (Figure 63) consists of listings of the data cards prepared by the analyst in accordance with the formats output by HAL. If there are any errors on these cards, HOS will output a diagnostic

```
                              DEMO PROGRAM -- RADAR PLOTTING                                  09/08/78.

INSTRM

   1    14400000141500120015    24224000000000000006    05252000150000000001    05252000120000000007    03000000550300000056     5
   2    03000000570200000062    05252000120000000006    05252000150000000003    05120000120000000000    05120200260000000000    10
   3    02000000620000000000    23240122245555555555    00000000000000000001    00000000000000000001    14300000240525200016    15
   4    00000000601410000027    05300000160000000000    00000000000200000063    02000000630000000000    15111040516045555555    20
   5    00000000000000000035    00000000000000000001    05160455555555555555    00000000000000000040    00000000000000000001    25
   6    14400000340300000041    02000000640200000064    23240122245555555555    00000000000000000063    00000000000000000001    30
   7    13001100001440000046    10400000111470000051    05120000210000000001    02000000651470000054    05252000100000000004    35
   8    05120000200000000000    02000000650200000065    23240122245555555555    00000000000000000076    00000000000000000001    40
   9    22010401225515170405    00000000000000000100    00000000000000000001    05162405225522010401    00000000000000000104    45
  10    00000000000000000001                                                                                                50

INSTRI

   1    01400000670200000066    02000000640000000000    01400000121500260014    07206000200000000004    14000000140406000026     5
   2    00000000250504300024    00000000410504000020    05040000211500260020    07206000400000000004    10000010000102000067    10
   3    02000000670000000000    05162405225555555555    00000000000000000011    00000000000000000001    05043000150000000001    15
   4    02000000700200000070    14000000310300000024    15002400100611640000    00000000240000000024    02000000710300000054    20
   5    05041000161500160012    31200010000000000000    00000000001000010001    02000000710000000000    03100503113555555555    25
   6    00000000000000000041    00000000000000000001    01600000240200000072    02000000720000000000    05040000170200000073    30

ISETSV

   6    7    3    1    7    4    2    5    1   20   11   14    1    2    3    4    5
```

Figure 62. Arrays Input to HOS from HAL.

```
                                         DEMO PROGRAM -- RADAR PLOTTING

SYSTEM                        DEMO PROGRAM -- RADAR PLOTTING
METRIC                      0  25  -50

DISPLAY SECTION
RADAR DISPLAY                25  0 0 0.5  0 26 28 OFF
RADAR SCALE                14 0 1 0.5 -10.23 30.3 43.25 32
RADAR CENTER               25 0 1 0.5 0 26 28 0 0

CONTROL SECTION
LOAD                        1 0 1 .5 -21.2 18.8 66 DUMMY
RADAR MODE                 23 0 0 0.5 -6.5 12.13 1.8 0
HOOK VERIFY                23 0 0 0.5 -10.5 0 0 0
ENTER RADAR CONTACT 23 0 0 0.5 -4 15.25 3.6 0
TRACK BALL                 20 0 1 0.5 0 0 0 0 0

SYMBOL SECTION
RADAR CONTACT STATUS 16 0 0 0.5 0 26 28 BLANK
RADAR CONTACT POSITI 18 0 0 0.5 0 26 28 0 0
HOOK                       16 0 1 0.5 0 26 28 ON
HOOK RADIUS                17 0 1 0.5 0 26 28 .125
HOOK POSITION              18 0 1 0.5 0 26 28 0 0

OPERATOR FUNCTIONS
TRACK BALL POSITION   2   0   1   1   .04   0
```

Figure 63. Crewstation Input Data.

```
                              DEMO PROGRAM -- RADAR PLOTTING

MODEL SPECIFICATIONS
1. ROTARY SWITCH A  7 2 0 0.04 3.0 3 0 .36
2. ROTARY SWITCH B  7 2 0 0.04 2.2 3 0 .36
3. ROTARY SWITCH C  7 2 0 0.04 1.8 3 0 .36
4. ROTARY SWITCH D  7 2 0 0.04 1.6 3 0 .36
5. ROTARY SWITCH E  7 2 0 0.04 1.5 3 0 .36
6. ROTARY SWITCH F  7 2 0 0.04 1.4 3 0 .36
7. ROTARY SWITCH G  7 2 0 0.04 1.2 3 0 .36
8. TOGGLE SWITCH    7 2 0 0.04 0.6 3 0 .26
9. VARIAC A         8 2 0 0.05 3.2 2 360 360 110
10. VARIAC B        8 2 0 0.05 2.7 2 360 360 110
11. VARIAC C        8 2 0 0.05 1.6 2 360 360 110
12. IDIOT LIGHT     5 1 0 0.04 0 3
13. INDICATOR LIGHT 5 1 0 0.04 0 3
14. NUMERIC DISPLAY 5 1 0 0.04 0 3
15. CONTINUOUS DISP  5 1 .02 .04 0 3
16. DISCRETE SYMBOL 5 1 0 0.04 0 3
17. CONTINUOUS SYMB   5 1 .02 .04 0 2
18. POSITIONAL SYMB   5 1 .02 .04 0 2
19. HAND MIKE       7 2 0 0.04 5.1 3 0 .5
20.TRACKBALL         9 3 .02 .04 4.4 2   360 360 71  10
21. FOOTSWITCH        7 4 0 0.04 7.2 3 0 3.5
22. PUSHBUTTON      7 2 0 0.04 2.2  3 0 .5
23. PUSHBUTTON      7 3 0 0.04 2.2  3 0 .5
24. THUMBWHEEL      7 2 0 0.04 0.3 3 1 1
25. SCREEN DISPLAY   5 1 .02 .04 0 2
26. PUSH TO TURN A  8 2 0 0.04 1.8 2 360 360 170
27. VARIAC D        8 2 0 0.05 1.8 2 360 360 110
28. PUSH TO TURN B  8 2 0 0.04 1 2 360 360 170
29. VARIAC F        8 2 0 0.05 1 2 360 360 170
30. FLIR GRIP SWITCH7 3 0 0.04 2.5 3 0 .26
31. REAL/DISC SYMBOL   5 1 0 .04 0 2
END OF MODEL SPEX

HUMAN OPERATOR SPEX
EYES                0  51  -22  0  75  92
HANDS               10.5  25  -50  -10.5  25  -50
FEET                32  55  -122  -32  55  -122
SHOULDERS           22  0  -16  -22  0  -16
HIPS                17  0  -75  -17  0  -75
END OF HUMAN SPEX
```

Figure 63. Crewstation Input Data (Cont.)

message immediately after each card that is in error. These errors will usually not prevent the simulation from beginning its execution, but they will ultimately result in problems that must be corrected in order to obtain an error-free simulation.

The simulation output itself (Figure 64) can be quite extensive and usually must be studied carefully in order to ensure that both the operator and the system are behaving as they should. If one or the other are not behaving properly, changes to the HOPROC instructions or to the input data will generally be required. The simulation output is subdivided into two major column headings -- one column listing the operator actions, the other column listing hardware actions. The current simulation time is printed at the left-hand edge of the page. Next to the simulation time are the actions being taken by the operator. These actions are shown indented under the procedure name that is currently being executed. Statement labels, IF, and ALTER statement numbers are also indicated so that the progression of actions through the procedures can be followed. In addition, whenever the operator absorbs or remembers the value of a device or function, the new estimated value is shown. Thus the analyst can readily see what the operator *believes* the value of a device or function to be throughout the simulation.

In the center of the page there is a column that indicates which body part the operator is using to accomplish the indicated action. This column can be used to check that the appropriate body parts are being used for the appropriate functions (i.e., that the crewstation data deck has been properly prepared). It can also be used to estimate the loading on each of the operator's channels.*

*However, the HODAC Channel Loading Report presents this information in a much more concise format.

```
                                   DEMO PROGRAM -- RADAR PLOTTING                                               09/08/78.
                                                                                        BODY
OPERATOR                                                                       |  RH LH RF LF  E    HARDWARE
--------                                                                       |  --------------    --------

       .00 MISSION
       .00 RADAR PLOT
       .00 *RADAR DISPLAY
       .00      ALTER    29
       .14      ABSORB LOAD                                                                    X
       .22          LOAD                  = DUMMY                                                   DUMMY
       .70      MANIPULATE LOAD                                                       X
                                                                                                    SLOAD
       .70 END *RADAR DISPLAY
       .70 RADAR PLOT
       .70      IF    6
       .84      ABSORB RADAR CONTACT 1 STAT                                                    X
       .96          RADAR CONTACT 1 STAT = BLANK                                                    BLANK
       .97      STEP ENTER
       .97      ALTER    12
       .97      ABSORB RADAR CONTACT 1 POSI                                                    X
      1.21          RADAR CONTACT 1 POSI  = (    -4.0,    -4.1)                                     (   -4.0,    -4.0)
      1.21      ALTER    14
      1.21 *HOOK VERIFY
      1.21 &HOOK POSITION
      1.21      STEP CHECK
      1.22      ABSORB HOOK POSITION                                                           X
      1.46          HOOK POSITION         = (      .0,      .0)                                     (     .0,      .0)
      1.46      IF    35
      1.46      ABSORB HOOK POSITION                                                           X
      1.50          HOOK POSITION         = (      .0,      .0)                                     (     .0,      .0)
      1.51      COMPUTE TRACK BALL POSITION
      1.58      ABSORB RADAR SCALE                                                             X
      1.82          RADAR SCALE           =     32.0                                                    32.0
      2.09      ABSORB TRACK BALL                                                  X
      2.33          TRACK BALL            = (      .0,      .0)                                     (     .0,      .0)
      2.37          TRACK BALL POSITION   =     -1.2
      2.37      ALTER    41
      2.37      ABSORB TRACK BALL                                                  X
      2.41          TRACK BALL            = (      .0,      .0)                                     (     .0,      .0)
      2.41      MANIPULATE TRACK BALL                                              X
      2.41      IF    42
                                                                                                    STRACK BALL
      2.83      IF    42
      2.84      STEP CHECK
      2.91      ABSORB HOOK POSITION                                                           X
      2.95          HOOK POSITION         = (    -4.0,    -4.1)                                     (   -4.0,    -4.1)
      2.95      IF    35
      2.95      ABSORB HOOK POSITION                                                           X
      2.99          HOOK POSITION         = (    -4.0,    -4.1)                                     (   -4.0,    -4.1)
      3.00 END &HOOK POSITION
      3.00 *HOOK VERIFY
      3.00 END *HOOK VERIFY
      3.00 RADAR PLOT
```

Figure 64. HOS simulation results.

The right-hand column indicates which hardware procedures are being executed in response to the operator's actions. It also shows the actual values of the devices being used by the operator whenever the operator absorbs or recalls any information.

A final type of output that can be obtained from HOS is shown in Figure 65. This report is generated whenever an active MILESTONE or ENDPOINT is encountered. The output lists all the major arrays that are being used by HOS -- e.g., the X, Y, and Z locations of all displays, controls, and symbols, their estimated, desired, and actual values, etc. It also lists the current values of all operator and hardware functions, the names of all the operator procedures currently on the active procedure list, the locations of each of the major body parts, what each body part is doint, and when it will complete whatever it is doing.

4.6.2 (A) Starting from a Checkpoint

When a MILESTONE instruction is executed, a checkpoint is generated. Execution can be restarted at the chekcpoint by including a CHECKPOINT card in the input stream. This card has the format:

CHECKPOINT checkpoint-number

Input cards can be entered after the CHECKPOINT card to modify any of the crewstation data.

4.6.3 (A) Bypassing Crewstation Input Processing

After the crewstation inputs have been processed once (whether or not there were errors in the inputs) the bulk of the input data processing can be bypassed by the use of the READ INPUTS card. If there are any corrections to the crewstation data, these can be entered immediately after the READ INPUTS card.

```
                    DEMO PROGRAM -- RADAR PLOTTING                       09/08/78.
                                ENDPOINT      0

   DICTIONARY ENTRIES    SI  XDIM     YDIM     ZDIM     LOWER    UPPER   XVALUE   YVALUE  DESIRED ESTIMATED ESTIME     HAB

10 RADAR DISPLAY          1     .00    20.08    -8.06     6.00     6.00     7.00      .00     6.00     7.00      .00      .00
11 RADAR SCALE            1   -4.01    21.77    -2.66    32.00    32.00    32.00      .00    32.00    32.00    33.64      .48
12 RADAR CENTER           1     .00    20.08    -8.06      .00      .00      .00      .00      .00      .00      .00      .00

13 LOAD                   1   -8.35    17.24     6.30     1.00     1.00     1.00      .00     1.00     1.00     1.06      .77
14 TRACK BALL             1     .00     9.64   -19.69     1.71     1.71     1.72      .00     1.71     1.71    35.18      .99
15 RADAR MODE             0   -2.56    14.62   -18.98      .00      .00      .00      .00      .00      .00     4.57      .00
16 HOOK VERIFY            0   -4.13     9.64   -19.69      .00      .00      .00      .00      .00      .00    35.57      .00
17 ENTER RADAR CONTACT    1   -1.57    15.85   -18.27      .00      .00      .00      .00      .00      .00    35.57      .00

18 HOOK                   1     .17    20.08    -8.06     7.00     7.00     7.00      .00     7.00     7.00      .00      .00
19 HOOK RADIUS            1     .17    20.08    -8.06      .13      .13      .13      .00      .13      .13      .00      .00
20 HOOK POSITION          1     .17    20.08    -8.06     5.27     5.77     5.51      .01     5.52     5.51    35.29     1.00
21 RADAR CONTACT          0     .00      .00      .00      .00      .00      .00      .00      .00      .00      .00      .00
22 RADAR CONTACT STATUS   0     .00      .00      .00      .00      .00      .00      .00      .00    21.00      .00      .00
23 RADAR CONTACT 1 STAT   1    -.12    20.08    -8.79     2.00     2.00     7.00      .00     2.00     4.00      .96      .75
24 RADAR CONTACT 2 STAT   1    -.09    20.08    -8.57     2.00     2.00     7.00      .00     2.00     4.00     4.87      .75
25 RADAR CONTACT 3 STAT   1     .00    20.08    -8.63     2.00     2.00     7.00      .00     2.00     4.00     9.52      .75
26 RADAR CONTACT 4 STAT   1     .09    20.08    -8.71     2.00     2.00     7.00      .00     2.00     4.00    13.90      .75
27 RADAR CONTACT 5 STAT   1     .12    20.08    -8.79     2.00     2.00     7.00      .00     2.00     4.00    16.74      .75
28 RADAR CONTACT 6 STAT   1    -.25    20.08    -8.00     2.00     2.00     7.00      .00     2.00     4.00    19.10      .75
29 RADAR CONTACT 7 STAT   1     .22    20.08    -8.45     2.00     2.00     7.00      .00     2.00     4.00    21.74      .75
30 RADAR CONTACT 8 STAT   1     .17    20.08    -8.72     2.00     2.00     7.00      .00     2.00     4.00    26.33      .75
31 RADAR CONTACT 9 STAT   1     .21    20.08    -8.61     2.00     2.00     7.00      .00     2.00     4.00    28.78      .75
32 RADAR CONTACT 10 STA   1     .17    20.08    -8.06     2.00     2.00     4.00      .00     2.00     4.00    31.34      .75
33 RADAR CONTACT POSITI   0     .00      .00      .00      .00      .00      .00      .00      .00    21.00      .00      .00
34 RADAR CONTACT 1 POSI   1    -.12    20.08    -8.79      .00      .00    -4.00    -4.00      .00    -3.97     1.21      .72
35 RADAR CONTACT 2 POSI   1    -.09    20.08    -8.57      .00      .00    -3.00     3.00      .00    -3.02     5.12      .72
36 RADAR CONTACT 3 POSI   1     .00    20.08    -8.63      .00      .00      .00     1.00      .00      .01     9.77      .72
37 RADAR CONTACT 4 POSI   1     .09    20.08    -8.71      .00      .00     3.00    -1.50      .00     2.96    14.15      .72
38 RADAR CONTACT 5 POSI   1     .12    20.08    -8.79      .00      .00     4.00    -4.00      .00     3.99    16.99      .72
39 RADAR CONTACT 6 POSI   1    -.25    20.08    -8.00      .00      .00    -8.00     2.00      .00    -7.99    19.35      .72
40 RADAR CONTACT 7 POSI   1     .22    20.08    -8.45      .00      .00     7.00    -6.00      .00     7.01    21.99      .72
41 RADAR CONTACT 8 POSI   1     .17    20.08    -8.72      .00      .00     5.50    -2.00      .00     5.53    26.58      .72
42 RADAR CONTACT 9 POSI   1     .21    20.08    -8.61      .00      .00     7.50     1.50      .00     7.47    29.03      .72
43 RADAR CONTACT 10 POS   1     .17    20.08    -8.06      .00      .00     5.50      .00      .00     5.52    31.59      .72
```

Figure 65. MILESTONE/ENDPOINT Output.

```
                         DEMO PROGRAM -- RADAR PLOTTING                  09/08/78.
                                   ENDPOINT      0

   DICTIONARY ENTRIES   ST  LOWER    UPPER   XVALUE   YVALUE  DESIRED   ESTIME      HAB

44 TRACK BALL POSITION   1      .00      .00     1.71     -.02      .00    34.73     1.00

    DICTIONARY ENTRIES      ACTUAL    ESTIME      STATE

45  TARGET MATRIX             .000      .697       .000
46  HOOK LIMITS               .250      .697       .000
47  RADAR SWEEP              1.000      .697       .000
48  NEW BALL POSITION        1.723    35.143       .000
49  HOOKED POSITION          5.500    35.568       .000

       ACTIVE PROCEDURES           CRIT    ESTIME      INT      LNT      NANY      TINT     STATE    RESULT

   54  MISSION                    1.000    35.568        0        2         0      .000     1.000      .000

EYES        CURRENTLY AT (    .17,  20.08,  -8.66)
WILL BE FREE AT TIME T =   35.21

RIGHT HAND CURRENTLY AT (  -1.57,  15.85, -18.27) = CONTROL ENTER RADAR CONTACT
WILL BE FREE AT TIME T=  36.07

LEFT HAND  CURRENTLY AT (  -4.13,   9.84, -19.69) = CONTROL HOOK VERIFY
WILL BE FREE AT TIME T=  36.07

RIGHT FOOT CURRENTLY AT (  12.60,  21.65, -48.03)
WILL BE FREE AT TIME T =      .00

LEFT FOOT  CURRENTLY AT ( -12.60,  21.65, -48.03)
WILL BE FREE AT TIME T =      .00
```

Figure 65. MILESTONE/ENDPOINT Output (Cont.)

The READ INPUTS card and CHECKPOINT card cannot be used together (the CHECKPOINT card has the effect of bypassing the crewstation input processing).

4.6.4 (A) HOS Run-Time Inputs

The hardware and operator functions normally accept run-time inputs from logical unit 7. Supplementary inputs can, however, be read in on logical unit 5 if the user precedes them by an extra $^{7}8_{9}$ card (to separate them from the CHECKPOINT, READ INPUTS, or other crewstation data cards that could potentially appear in the input stream at this point).

4.7 STEPS HALHOD AND HODAC

Step HALHOD uses the data supplied by HAL to prepare the HODAC Analysis Program for execution. Step HODAC executes the HODAC program. Since one of the HODAC analyses, the Devices By Procedure Analysis, requires a different version of the HODAC program than the other analyses, steps HALHOD and HODAC have to be executed twice to obtain all the available HODAC analyses. The following control cards are used to obtain any analysis other than the Devices By Procedure Analysis:

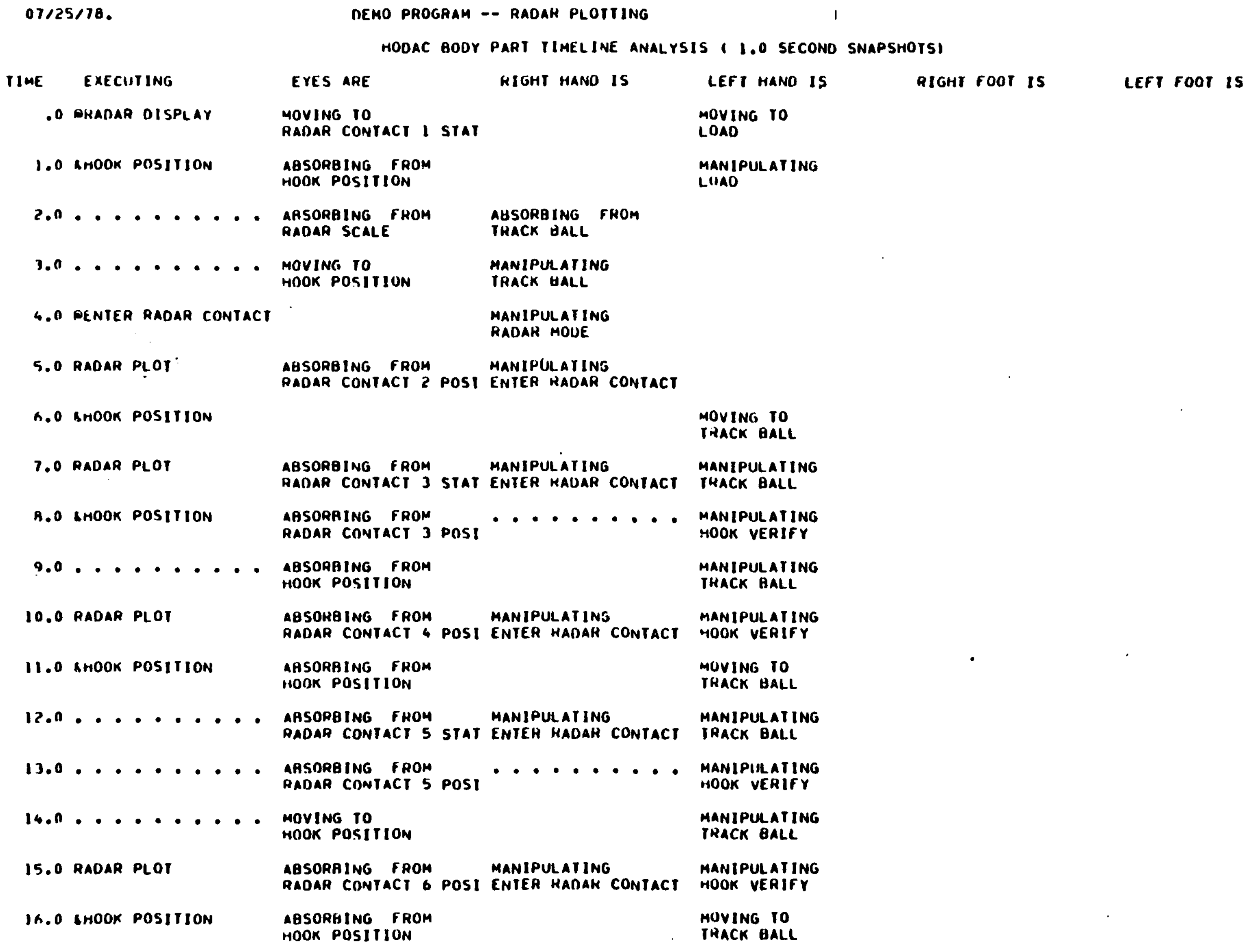
```
  CCEXEC, HOSEXEC, HODAC.    NAME = name, RUN = 0
CCEXEC, HOSEXEC, HALHOD.   NAME = name, RUN = 0
```

The following control cards are used to obtain the Devices By Procedure Analysis:

```
  CCEXEC, HOSEXEC, HODAC.    NAME = name, RUN = 1
CCEXEC, HOSEXEC, HALHOD.   NAME =- name, RUN = 1
```

4.7.1 The Timeline Analysis (Figure 66)

The HODAC Timeline Analysis provides a summary of the primary actions performed by the operator within a specific interval, termed a



```
07/25/78.                    DEMO PROGRAM -- RADAR PLOTTING

                              HODAC BODY PART TIMELINE ANALYSIS ( 1.0 SECOND SNAPSHOTS)

TIME    EXECUTING            EYES ARE             RIGHT HAND IS        LEFT HAND IS         RIGHT FOOT IS        LEFT FOOT IS

   .0 @RADAR DISPLAY         MOVING TO                                 MOVING TO
                             RADAR CONTACT 1 STAT                      LOAD

  1.0 &HOOK POSITION         ABSORBING  FROM                           MANIPULATING
                             HOOK POSITION                             LOAD

  2.0 . . . . . . . . . . .  ABSORBING  FROM      ABSORBING  FROM
                             RADAR SCALE          TRACK BALL

  3.0 . . . . . . . . . . .  MOVING TO            MANIPULATING
                             HOOK POSITION        TRACK BALL

  4.0 @ENTER RADAR CONTACT                        MANIPULATING
                                                  RADAR MODE

  5.0 RADAR PLOT             ABSORBING  FROM      MANIPULATING
                             RADAR CONTACT 2 POSI ENTER RADAR CONTACT

  6.0 &HOOK POSITION                                                   MOVING TO
                                                                       TRACK BALL

  7.0 RADAR PLOT             ABSORBING  FROM      MANIPULATING         MANIPULATING
                             RADAR CONTACT 3 STAT ENTER RADAR CONTACT  TRACK BALL

  8.0 &HOOK POSITION         ABSORBING  FROM      . . . . . . . . . .  MANIPULATING
                             RADAR CONTACT 3 POSI                      HOOK VERIFY

  9.0 . . . . . . . . . . .  ABSORBING  FROM                           MANIPULATING
                             HOOK POSITION                             TRACK BALL

 10.0 RADAR PLOT             ABSORBING  FROM      MANIPULATING         MANIPULATING
                             RADAR CONTACT 4 POSI ENTER RADAR CONTACT  HOOK VERIFY

 11.0 &HOOK POSITION         ABSORBING  FROM                           MOVING TO
                             HOOK POSITION                             TRACK BALL

 12.0 . . . . . . . . . . .  ABSORBING  FROM      MANIPULATING         MANIPULATING
                             RADAR CONTACT 5 STAT ENTER RADAR CONTACT  TRACK BALL

 13.0 . . . . . . . . . . .  ABSORBING  FROM      . . . . . . . . . .  MANIPULATING
                             RADAR CONTACT 5 POSI                      HOOK VERIFY

 14.0 . . . . . . . . . . .  MOVING TO                                 MANIPULATING
                             HOOK POSITION                             TRACK BALL

 15.0 RADAR PLOT             ABSORBING  FROM      MANIPULATING         MANIPULATING
                             RADAR CONTACT 6 POSI ENTER RADAR CONTACT  HOOK VERIFY

 16.0 &HOOK POSITION         ABSORBING  FROM                           MOVING TO
                             HOOK POSITION                             TRACK BALL
```

Figure 66. HODAC timeline analysis.

snapshot interval. At the top of the page, HODAC lists the date, the name of the simulation, and the snapshot interval. Across the page are headings for the simulation time, the procedure currently being executed, and each body part. Down the page are the simulation times, the primary procedure being executed within that snapshot interval, and the primary functions being performed by each body part during the snapshot interval. Note that the primary function is defined as that function requiring most of the operator's time during that snapshot interval.

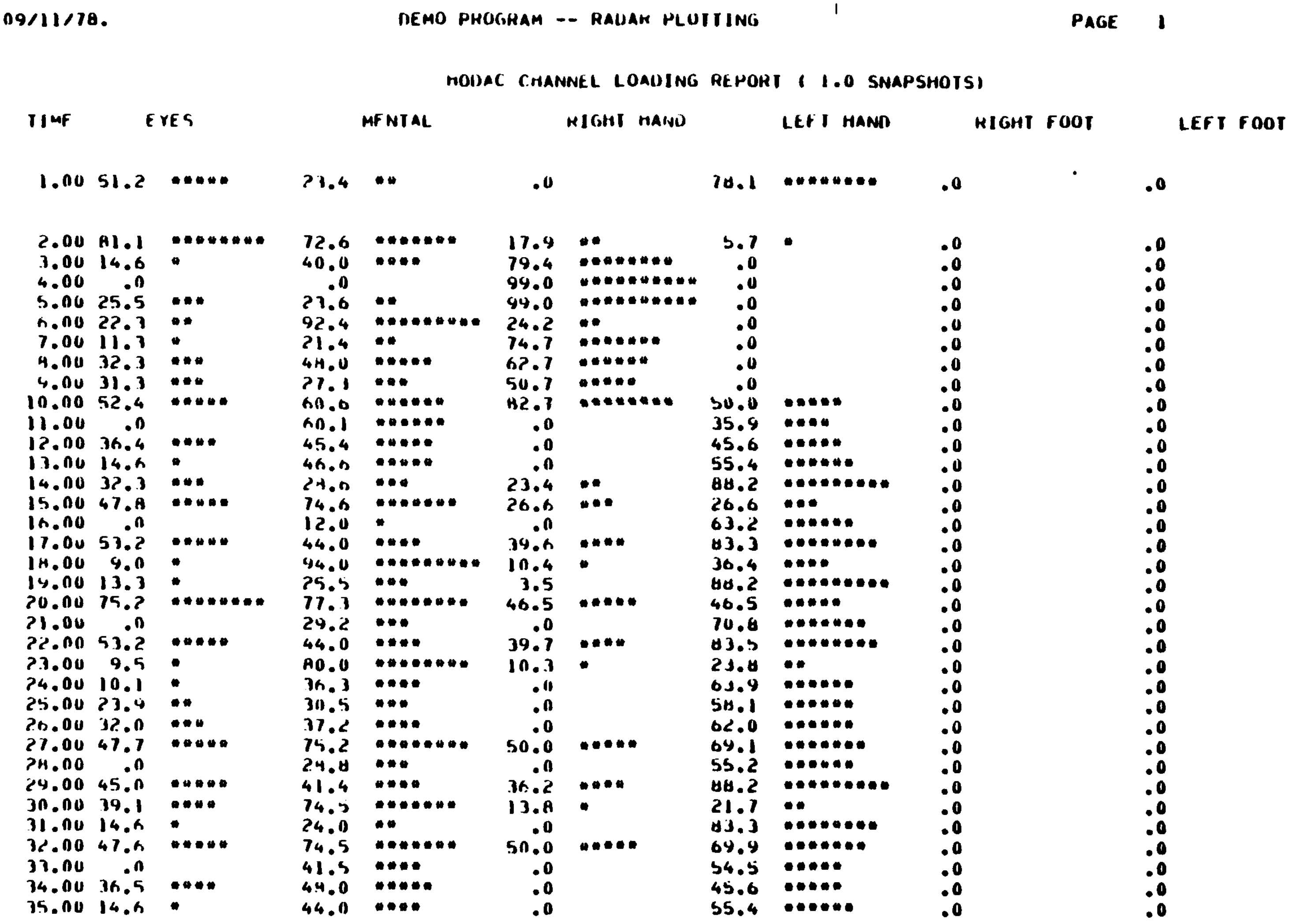
4.7.2 The Channel Loading Report

The Timeline Analysis identifies the primary actions performed by the operator within each snapshot interval. Examining the report can help to determine how busy the operator is at any particular time. However, since the Timeline Analysis identifies the primary activity performed in any snapshot interval, it does not indicate how much of that interval was actually spent performing that activity (or any activity other than the primary activity). This data is provided by the Channel Loading Report. This report (Figure 67) indicates the percent of time that each body part spends on any activity within a specific snapshot interval.

4.7.3 The Device By Body Parts Analysis

The Device Analysis by Body Parts is closely related to the Timeline Analysis and the Channel Loading Analysis. The report (Figure 68) tabulates statistics on the amount of time each of the operator's body parts spends performing certain types of tasks related to each device. The name of the device is listed in the leftmost column. Each device generates three lines of output. The first line contains the times associated with moving to and, if necessary, grasping the device. The second line contains the times associated with absorbing the value of the device. The third line contains the amounts of times associated with manipulating the device. The times are listed underneath the column headed by the appropriate body part.*

*The "manipulation" times listed under "EYES" column is actually the amount of time spent recalling the value of the device, since the operator's eyes cannot be used in manipulations.



```
09/11/78.                      DEMO PROGRAM -- RADAR PLOTTING                         PAGE   1

                                    HODAC CHANNEL LOADING REPORT ( 1.0 SNAPSHOTS)

TIME     EYES               MENTAL             RIGHT HAND         LEFT HAND          RIGHT FOOT     LEFT FOOT

 1.00  51.2  *****        23.4  **             .0               78.1  ********       .0             .0


 2.00  81.1  ********     72.6  *******      17.9  **            5.7  *              .0             .0
 3.00  14.6  *            40.0  ****         79.4  ********       .0                 .0             .0
 4.00    .0                 .0               99.0  **********     .0                 .0             .0
 5.00  25.5  ***          23.6  **           99.0  **********     .0                 .0             .0
 6.00  22.3  **           92.4  *********    24.2  **             .0                 .0             .0
 7.00  11.3  *            21.4  **           74.7  *******        .0                 .0             .0
 8.00  32.3  ***          48.0  *****        62.7  ******         .0                 .0             .0
 9.00  31.3  ***          27.1  ***          50.7  *****          .0                 .0             .0
10.00  52.4  *****        60.6  ******       82.7  ********     50.0  *****          .0             .0
11.00    .0               60.1  ******         .0               35.9  ****           .0             .0
12.00  36.4  ****         45.4  *****          .0               45.6  *****          .0             .0
13.00  14.6  *            46.6  *****          .0               55.4  ******         .0             .0
14.00  32.3  ***          28.6  ***          23.4  **           88.2  *********      .0             .0
15.00  47.8  *****        74.6  *******      26.6  ***          26.6  ***            .0             .0
16.00    .0               12.0  *              .0               63.2  ******         .0             .0
17.00  53.2  *****        44.0  ****         39.6  ****         83.3  ********       .0             .0
18.00   9.0  *            94.0  *********    10.4  *            36.4  ****           .0             .0
19.00  13.3  *            25.5  ***           3.5               88.2  *********      .0             .0
20.00  75.2  ********     77.3  ********     46.5  *****        46.5  *****          .0             .0
21.00    .0               29.2  ***            .0               70.8  *******        .0             .0
22.00  53.2  *****        44.0  ****         39.7  ****         83.5  ********       .0             .0
23.00   9.5  *            80.0  ********     10.3  *            23.8  **             .0             .0
24.00  10.1  *            36.3  ****           .0               63.9  ******         .0             .0
25.00  23.9  **           30.5  ***            .0               58.1  ******         .0             .0
26.00  32.0  ***          37.2  ****           .0               62.0  ******         .0             .0
27.00  47.7  *****        75.2  ********     50.0  *****        69.1  *******        .0             .0
28.00    .0               28.8  ***            .0               55.2  ******         .0             .0
29.00  45.0  *****        41.4  ****         36.2  ****         88.2  *********      .0             .0
30.00  39.1  ****         74.5  *******      13.8  *            21.7  **             .0             .0
31.00  14.6  *            24.0  **             .0               83.3  ********       .0             .0
32.00  47.6  *****        74.5  *******      50.0  *****        69.9  *******        .0             .0
33.00    .0               41.5  ****           .0               54.5  *****          .0             .0
34.00  36.5  ****         49.0  *****          .0               45.6  *****          .0             .0
35.00  14.6  *            44.0  ****           .0               55.4  ******         .0             .0
```

Figure 67. HODAC Channel loading report.

```
07/25/78.                      DEMO PROGRAM -- RADAR PLOTTING

                                HODAC DEVICE ANALYSIS BY BODY PART

   DEVICE                   EYES                RIGHT HAND            LEFT HAND             RIGHT FOOT            LEFT FOOT

                      SUM   N   AVG   SD    SUM   N   AVG   SD    SUM   N   AVG   SD    SUM   N   AVG   SD    SUM   N   AVG   SD

 RADAR SCALE
MOVING/GRASPING        .07/  1= .071 .000
ABSORBING-COMPUTING    .32/  1= .321 .000
MANIPULATION-RECALL   2.16/  9= .241 .000

 LOAD
MOVING/GRASPING        .14/  1= .141 .000                        .48/  1= .481 .000
ABSORBING-COMPUTING    .12/  1= .121 .000
MANIPULATION-RECALL                                              .36/  1= .361 .000

 RADAR MODE
MOVING/GRASPING                              .30/  1= .301 .000
ABSORBING-COMPUTING
MANIPULATION-RECALL                          .50/  1= .501 .000

 HOOK VERIFY
MOVING/GRASPING                              .27/  1= .271 .000  2.41/  9= .271 .000
ABSORBING-COMPUTING
MANIPULATION-RECALL                          .50/  1= ;501 .000  4.00/  9= .441 .157

 ENTER RADAR CONTACT
MOVING/GRASPING                              .16/  1= .161 .000
ABSORBING-COMPUTING
MANIPULATION-RECALL                         4.50/ 10= .451 .150

 TRACK BALL
MOVING/GRASPING                              .27/  1= .271 .000  2.41/  9= .271 .000
ABSORBING-COMPUTING                          .36/  2= .181 .140
MANIPULATION-RECALL   4.44/ 18= .251 .027    .40/  1= .401 .000  3.61/  9= .401 .000

 RADAR CONTACT
MOVING/GRASPING        .31/ 10= .031 .036
ABSORBING-COMPUTING   5.60/ 20= .281 .040
MANIPULATION-RECALL

 RADAR CONTACT STATUS
MOVING/GRASPING        .31/ 10= .031 .036
ABSORBING-COMPUTING   2.40/ 10= .241 .000
MANIPULATION-RECALL

 RADAR CONTACT 1 STAT
MOVING/GRASPING        .14/  1= .141 .000
ABSORBING-COMPUTING    .24/  1= .241 .000
MANIPULATION-RECALL

 RADAR CONTACT 2 STAT
MOVING/GRASPING        .02/  1= .021 .000
ABSORBING-COMPUTING    .24/  1= .241 ,000
  IPULATION-RECALL
```

Figure 68. HODAC device by body part analysis.

Statistics are presented in a format that is standard for most of the remaining HODAC reports. The format gives the total amount of time spent on the activity, the number of times the activity was performed, the mean value for the activity, and the standard deviation. Thus, as shown in Figure 68 , the operator spends a total of 4.5 seconds manipulating the ENTER-RADAR-CONTACT control with his right hand. He performs 10 manipulations for an average manipulation time of .45 seconds with a standard deviation of .15 seconds.

4.7.4 The Device Analysis By Usage (Figure 69)

The Device By Usage Analysis generates statistics on the amounts of time spent by the operator in specific actions associated with each of the devices in the crewstation. It thus provides composite statistics for the actions presented in the devices by body part analysis and, in addition, provides statistics on activities that may have required several body parts -- for example, enabling a display or control. The format of the report is similar to that of the Device By Body Parts Analysis. The left-most column identifies the device. The remaining columns list the total times, number of times, means and standard deviations for each of the usages identified at the TOP of the page. The meanings of each of these usages are as follows:

Moving/Grasping	The time spent moving a body part to a device and (if required) grasping it.
Absorbing/Computing	The time spent reading a device or computing a function.
Manipulating	The time spent manipulating a control.
Recalling	The time spent recalling (or attempting to recall) the value of a device or function.
Enabling	The time spent working on the procedure that enables the device or function.

```
07/25/78.                         DEMO PROGRAM -- RADAR PLOTTING

                                   HODAC DEVICE ANALYSIS BY USAGE

DEVICE               MOVING/GRASPING     ABSORBING-COMPUTING  MANIPULATING        RECALLING           ENABLING

RADAR DISPLAY                                                                                           .74/  1= .741 .000

RADAR SCALE            .07/  1= .071 .000   .32/  1= .321 .000                     2.16/  9= .241 .000

LOAD                   .62/  2= .311 .170   .12/  1= .121 .000   .36/  1= .361 .000

RADAR MODE             .30/  1= .301 .000                        .50/  1= .591 .000

HOOK VERIFY           2.68/ 10= .271 .000                       4.50/  9= .501 .000                       .00/ 20= .001 .000

ENTER RADAR CONTACT    .16/  1= .161 .000                       4.50/  9= .501 .000                       .81/  1= .811 .000

TRACK BALL            2.68/ 10= .271 .000   .36/  2= .181 .140  4.02/ 10= .401 .000  4.44/ 18= .251 .027

RADAR CONTACT          .31/ 10= .031 .036  5.60/ 20= .281 .040

RADAR CONTACT STATUS   .31/ 10= .031 .036  2.40/ 10= .241 .000

RADAR CONTACT 1 STAT   .14/  1= .141 .000   .24/  1= .241 .000

RADAR CONTACT 2 STAT   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT 3 STAT   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT 4 STAT   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT 5 STAT   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT 6 STAT   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT 7 STAT   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT 8 STAT   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT 9 STAT   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT 10 STA   .02/  1= .021 .000   .24/  1= .241 .000

RADAR CONTACT POSITI                       3.20/ 10= .321 .000

RADAR CONTACT 1 POSI                        .32/  1= .321 .000

RADAR CONTACT 2 POSI                        .32/  1= .321 .000

RADAR CONTACT 3 POSI                        .32/  1= .321 .000

RADAR CONTACT 4 POSI                        .32/  1= .321 .000

RADAR CONTACT 5 POSI                        .32/  1= .321 .000

RADAR CONTACT 6 POSI                        .32/  1= .321 .000
```

Figure 69. HODAC device by usage analysis.

Adjusting — The time spent working on the procedure that adjusts the device or function.

Disabling — The time spent working on the procedure that disables the device or function.

Stymied-Absorption — The time spent outside the current procedure as a consequence of being unable to read the value of a device at the time it was needed.

Stymied-Manipulation — The time spent outside the current procedure as a consequence of being unable to manipulate the device when needed.

4.7.5 The Device By Procedure Analysis (Figure 70)

The Device By Procedure Analysis summarizes the times spent by the operator on the same actions as in the Device By Usage Analysis. The difference between the two is that in the Devices By Procedures Analysis, the actions are broken down by procedure, as well as by usage.

4.7.6 The Procedures Analysis (Figure 71)

The Procedures Analysis also presents usage statistics. However, in this case, the data is summed over all actions performed within a procedure rather than over procedures. In addition, the Procedures Analysis accumulates statistics on certain types of activities that only have significance at the procedural level.

4.7.7 Label Analysis (Figure 72)

The HODAC Label Analysis provides certain summary data for procedures and, in addition, tabulates data pertaining to certain procedure activities. The summary data for each procedure includes the times at which procedures were first and last activated, the times at which they were first and last executed, and the times at which they were first and last removed from the active procedure list.

```
09/08/78.                  DEMO PROGRAM -- RADAR PLOTTING                 PAGE   2
                             MODAC DEVICE ANALYSIS BY PROCEDURE
                                   PROCEDURE RADAR PLOT
DEVICE               MOVING/GRASPING      ABSORBING-COMPUTING   MANIPULATING        RECALLING          ENABLING
RADAR DISPLAY                                                                                           .70/  1= .70I .000
RADAR MODE                                                       9.54/ 18= .53I .136
HOOK VERIFY            2.41/  9= .27I .000                       4.50/  9= .50I .000                   26.65/ 10=2.67I .957
ENTER RADAR CONTACT     .40/  2= .24I .086                       4.50/  9= .50I .000                     .01/  1= .01I .000
RADAR CONTACT           .31/ 10= .03I .036   3.60/ 20= .18I .060
RADAR CONTACT STATUS    .31/ 10= .03I .036   1.20/ 10= .12I .000
RADAR CONTACT 1 STAT    .14/  1= .14I .000    .12/  1= .12I .000
RADAR CONTACT 2 STAT    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT 3 STAT    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT 4 STAT    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT 5 STAT    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT 6 STAT    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT 7 STAT    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT 8 STAT    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT 9 STAT    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT 10 STA    .02/  1= .02I .000    .12/  1= .12I .000
RADAR CONTACT POSITI                         2.40/ 10= .24I .000
RADAR CONTACT 1 POSI                          .24/  1= .24I .000
RADAR CONTACT 2 POSI                          .24/  1= .24I .000
RADAR CONTACT 3 POSI                          .24/  1= .24I .000
RADAR CONTACT 4 POSI                          .24/  1= .24I .000
RADAR CONTACT 5 POSI                          .24/  1= .24I .000
RADAR CONTACT 6 POSI                          .24/  1= .24I .000
RADAR CONTACT 7 POSI                          .24/  1= .24I .000
RADAR CONTACT 8 POSI                          .24/  1= .24I .000
RADAR CONTACT 9 POSI                          .24/  1= .24I .000
```

Figure 70. Devices by ɔcedure analysis.

```
 07/25/78.                          DEMO PROGRAM -- RADAR PLOTTING

                                         HODAC PROCEDURAL ANALYSIS

PROCEDURE               MOVING/GRASPING     ABSORBING           MANIPULATING        RECALLING           ENABLING

RADAR PLOT               3.14/ 21= .15% .118  5.60/ 20= .28% .040  9.00/ 18= .50% .000                      1.54/ 22= .07% .222
*RADAR DISPLAY            .62/  2= .31% .170   .12/  1= .12% .000   .36/  1= .36% .000
*HOOK POSITION           3.17/ 31= .10% .115  2.56/ 43= .06% .071  4.02/ 10= .40% .000  6.60/ 27= .24% .023
*ENTER RADAR CONTACT      .30/  1= .30% .000                        .50/  1= .50% .000
```

Figure 71. HODAC procedures analysis.

```
07/25/78.                        DEMO PROGRAM -- RADAR PLOTTING

                                           HODAC LABEL ANALYSIS


MISSION                  FIRST ACTIVATED      .00     LAST ACTIVATED
                         FIRST EXECUTED       .00     LAST EXECUTED     27.93
                         FIRST REMOVED      27.93     LAST REMOVED

RADAR PLOT               FIRST ACTIVATED      .00     LAST ACTIVATED
                         FIRST EXECUTED       .00     LAST EXECUTED     27.67
                         FIRST REMOVED      27.93     LAST REMOVED

                 LABEL     TOTAL TIME TO:             ACTIVE TIME TO:             NUMBER OF ENCOUNTERS:    ENCOUNTERS

                 ENTER         .39/  1=   .39:   .00      .39/  1=   .39:   .00   10/  1= 10.00            1/  1=100.00

*RADAR DISPLAY           FIRST ACTIVATED      .00     LAST ACTIVATED
                         FIRST EXECUTED       .00     LAST EXECUTED       .00
                         FIRST REMOVED        .74     LAST REMOVED

*HOOK POSITION           FIRST ACTIVATED     1.44     LAST ACTIVATED    26.01
                         FIRST EXECUTED      1.44     LAST EXECUTED     26.01
                         FIRST REMOVED       3.45     LAST REMOVED      27.67

                 LABEL     TOTAL TIME TO:             ACTIVE TIME TO:             NUMBER OF ENCOUNTERS:    ENCOUNTERS

                 CHECK         .00/ 10=   .00:   .00      .00/ 10=   .00:   .00   20/ 10=  2.00            10/ 10=100.00

*HOOK VERIFY             FIRST ACTIVATED     1.44     LAST ACTIVATED    26.01
                         FIRST EXECUTED      1.44     LAST EXECUTED     27.67
                         FIRST REMOVED       3.45     LAST REMOVED      27.67

*ENTER RADAR CONTACT     FIRST ACTIVATED     3.72     LAST ACTIVATED
                         FIRST EXECUTED      3.72     LAST EXECUTED       .00
                         FIRST REMOVED       4.53     LAST REMOVED
```

Figure 72. HODAC label analysis.

The within-procedure times are predicated on the assumption that labeled statements within a procedure identify significant blocks of code within the procedure. Consequently, the data accumulated in association with these labels describe how frequently these blocks of code within a procedure are executed and how long each execution requires. The column entitled "TOTAL TIME TO" gives the amount of time, number of times, mean and standard deviations that it took between the time a procedure was activated and the time that the label was encountered. The column headed "ACTIVE TIME TO" gives the total amount of time, number of times, mean and standard deviation between the time the procedure began execution, and tne time the labeled statement was encountered. Consequently, these columns will always contain, at most, a single count for each activation of a procedure.

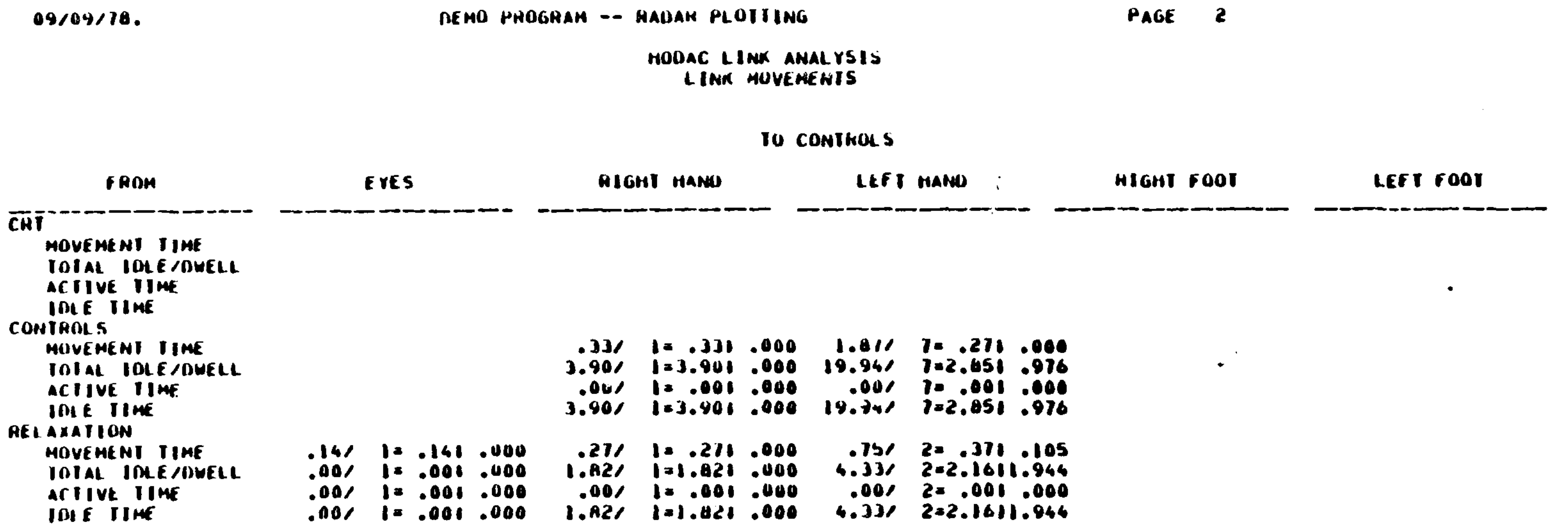
The column headed "NUMBER OF ENCOUNTERS" tabulates the number of times per execution that the labeled statement was encountered. This number is indicative of the frequency with which a particular section of code was executed. Finally, the last column, "ENCOUNTERS," indicates the number and percent of times the labeled statement was executed for each distinct execution of the procedure.

These statistics can have important consequences when one attempts to determine the training requirements for a specific system. In particular, they indicate how much concentration should be placed on training operators on certain routine operations.

4.7.8 The Link Analysis (Figures 73 and 74)

In the HODAC Link Analysis, the analyst defines groups of displays and controls. HODAC will then accumulate statistics on the number of operator "transitions" from group to group and within each group. These statistics include:

- Times for moving a body part from an element in one group to an element in another group.



```
09/09/78.                  DEMO PROGRAM -- RADAR PLOTTING                         PAGE   2

                                        HODAC LINK ANALYSIS
                                          LINK MOVEMENTS

                                                        TO CONTROLS

       FROM                 EYES                RIGHT HAND            LEFT HAND             RIGHT FOOT            LEFT FOOT
 ---------------     ---------------     ---------------     ---------------     ---------------     ---------------
CRT
   MOVEMENT TIME
   TOTAL IDLE/DWELL
   ACTIVE TIME
   IDLE TIME
CONTROLS
   MOVEMENT TIME                             .33/  1= .33| .000   1.87/  7= .27| .000
   TOTAL IDLE/DWELL                         3.90/  1=3.90| .000  19.94/  7=2.85| .976
   ACTIVE TIME                               .00/  1= .00| .000    .00/  7= .00| .000
   IDLE TIME                                3.90/  1=3.90| .000  19.94/  7=2.85| .976
RELAXATION
   MOVEMENT TIME         .14/  1= .14| .000   .27/  1= .27| .000    .75/  2= .37| .105
   TOTAL IDLE/DWELL      .00/  1= .00| .000  1.82/  1=1.82| .000   4.33/  2=2.16||.944
   ACTIVE TIME           .00/  1= .00| .000   .00/  1= .00| .000    .00/  2= .00| .000
   IDLE TIME             .00/  1= .00| .000  1.82/  1=1.82| .000   4.33/  2=2.16||.944
```

Figure 73. HODAC link analysis.

```
09/09/78.                    DEMO PROGRAM -- RADAR PLOTTING                          PAGE   4

                                            HODAC LINK ANALYSIS
                                             LINK FREQUENCIES

                               EYES(   39)      R. HAND(     2)     L. HAND(    10)    R. FOOT(     0)    L. FOOT(     0)
                               ------------     ------------        ------------       ------------       ------------
CRT
     ALL   CRT                  37 / 94.872%

     TO    CONTROLS
     FROM  CONTROLS              1 /  2.564%
     ALL   CONTROLS              1 /  2.564%

     TO    RELAXATION
     FROM  RELAXATION
     ALL   RELAXATION

CONTROLS
     ALL   CONTROLS                                1 / 50.000%        7 / 70.000%

     TO    CRT                   1 /  2.564%
     FROM  CRT
     ALL   CRT                   1 /  2.564%

     TO    RELAXATION                                                 1 / 10.000%
     FROM  RELAXATION            1 /  2.564%       1 / 50.000%        2 / 20.000%
     ALL   RELAXATION            1 /  2.564%       1 / 50.000%        3 / 30.000%

RELAXATION
     ALL   RELAXATION

     TO    CRT
     FROM  CRT
     ALL   CRT

     TO    CONTROLS              1 /  2.564%       1 / 50.000%        2 / 20.000%
     FROM  CONTROLS                                                   1 / 10.000%
     ALL   CONTROLS              1 /  2.564%       1 / 50.000%        3 / 30.000%
```

Figure 74. HODAC link analysis.

- Total times that a body part is idle and/or dwelling in a particular group.

- Link frequencies (i.e., the percentage of all movements of a body part that are within a particular group or are from one particular group to another).

4.7.9 Inputs to HODAC

The only inputs required by HODAC are the control cards that invoke the desired analyses. The allowable syntax for these cards is described in detail in the HOS Users' Guide and on the HOS Reference Card. However, unless special constraints are to be placed on the analyses, the following control cards can be used to obtain each of the major HODAC analyses:

```
LABELS.
DEVICES BY PARTS.
DEVICES BY USAGE.
DEVICES BY PROCEDURE.
PROCEDURES.
```

A Timeline Analysis can be obtained by simply inserting the clause:

```
TIMELINE EVERY n SECONDS
```

prior to the period on any of the preceding control cards, where n is replaced by the "snapshot" interval.

A Channel Loading Report can be obtained by inserting a clause:

```
CHANNEL-LOADING EVERY n SECONDS
```

prior to the period on any of the preceding control cards, where n is replaced by the "snapshot" interval. (The Timeline clause and the Channel Loading clause cannot be used together.)

A Link Analysis requires the definition of the groups to be linked together in the analysis. The format for invoking a Link Analysis is:

```
LINKS       SYSTEM   system-name = device THRU device; device ...;
               .
               .
               .
            SYSTEM   system-name = device THRU device.
```

where system-name is an arbitrary (and optional) name supplied by the user and the device names are the names of the specific displays, controls, and/or symbols that are to be considered members of a group.

4.8 UTILITIES

Several utilities are available that provide listings for each of the HOS programs. These utilities -- LISTHAL, LISTHOS, and LISTHOD -- require no inputs.

This page intentionally
left blank.

APPENDIX A
THE HOS OPERATOR MODELS

APPENDIX A
THE HOS OPERATOR MODELS

A.1 COMPONENT MODELS USED IN HOS

The Human Operator Simulator is actually a set of models for a variety of components of human performance that have been integrated into a general composite model of the human operator. The HOS user must provide a precise description of the operator's task and the equipment that he must use to perform the task, together with specifications for several parameters relating to the operator's physical and performance characteristics. HOS then applies the various performance models to the user's input in order to produce, as output, a detailed description of the operator's performance in time.

The models embodied in HOS can be classified as follows:

- Display and control taxonomy.
- Task component taxonomy.
- Procedure multiplexing model.
- Anatomy movement models.
- Visual and tactile perception models.
- Memory model.
- Mental computation model.

The display and control taxonomy and task taxonomy are models that influence and must be evaluated separately from the dynamic performance models for which they structure the inputs and outputs. The procedure multiplexing model is the protocol by which the simulated operator (a serial processor) schedules the execution of his various tasks (procedures). These three models have a rather diffuse impact on HOS simulations in that it is extremely difficult to evaluate any of them except by assessing the adequacy of the total HOS system. These models and how they evolved are described in Sections A.2 through A.4.

The anatomy movement models describe the functional assignments of body parts (e.g., which hand to use to grasp a particular contro) and the dynamic response characteristics for each body part. A discussion of these models is presented in Section A.5.

Section A.6 presents a detailed description of the structure of the perception, memory, and mental computation models. These are the models of the operator's cognitive processes as represented in HOS. They describe how the operator functions as an information processor, i.e., how he obtains estimates for the values of the displays and controls, and performs mental calculations that are mathematical and logical functions of these values.

We will frequently point to theoretical and empirical support for the HOS models in the literature of human performance psychology. However, it should be remembered that these models have been developed to serve an immediate practical purpose -- to develop a simulation of the human operator that provides an economical tool for determining the human performance consequences of any particular man-machine interface design. Thus, we have had to impose practical constraints on the models in order not to overwhelm the information processing capacity of the computer and to maintain consistency within the HOS models. The main differences between the HOS models and the models in the psychological literature lie in the fact that we have developed several models for performance for which we could find no directly relevant published studies. For example, in the development of the procedure multiplexing model and the mental computation model, introspection and intuition have guided the model development. Since it was anticipated that the HOS mdoels would be improved as appropriate research findings became available, the HOS software system has been designed in a modular form to allow easy modification to any of the component models.

A.2 THE DEVICE TAXONOMY

The classification of devices used in HOS has a significant impact on all of the other HOS models. The classes of devices, as defined in HOS, are listed and described in Table A-1. This taxonomy places devices in different classes whenever qualitatively different procedures on the part of the operator would be required to use the devices. Within each class, differences between devices, such as size, location, and range of possible values, are specified by the user on HOS data cards. These data are used when a specific device is accessed in a specific situation.

A.3 THE TASK-COMPONENT TAXONOMY

The basic units of work in which HOS describes the actions of the simulated operator comprise a normative model of human performance. Two levels in the human performance hierarchy are employed in HOS -- the procedural level and the individual action level.

HOS procedures can vary from the "macro-level," those describing mission objectives, to the "micro-level," those describing detailed actions. There are five basic procedural actions. They are:

(1) Activate a procedure -- Place the procedure on an active procedure list from which it will be chosen for execution by the multiplexor when its criticality exceeds that of all other procedures on the list.

(2) Complete a procedure -- Discontinue execution of the current procedure until the specified procedure has completed execution.

Table A-1. HOS Device Taxonomy

DEVICE CLASS	DESCRIPTION
Discrete Displays	Devices which present information to the operator in terms of discrete settings.
Continuous Displays	Devices which present information to the operator in terms of values which may vary continuously along a single dimensional scale.
Positional Displays	Devices which present information to the operator in terms of ordered pairs of numbers.
Discrete Controls	Manipulable devices that can assume only discrete settings.
Continuous Controls	Manipulable devices that can assume values which may vary continuously along a single dimensional scale.
Positional Controls	Manipulable devices for which the device value is defined as the location of the control element.
Symbols	Figures that appear on CRT displays for which information is conveyed by the location of the figure on the screen and for which there may be a variety of associated attributes (e.g., size, color, type), each of which may be either discrete or continuous.

(3) Perform a procedure -- Place the procedure on the active procedure list and begin execution of the procedure immediately; do not resume execution of the current procedure until the specified procedure has completed execution.

(4) Monitor a device -- Place the procedure for adjusting the device on the active procedure list and periodically execute the adjust procedure until told to stop.

(5) End a procedure -- Remove the procedure (regardless of type) from the active procedure list.

These procedural actions have direct HOPROC statement counterparts, enabling complex procedural descriptions to be achieved with a minimum number of statement types.

The operator's repertoire of actions has been adapted principally from the studies of work movements performed by time-and-motion analysts (see, for example, Karger and Bayha, 1966). The major departure from the time-and-motion study taxonomy lies in the inclusion in HOS of a work unit for the process of mental computation. This addition was necessary in order to enable HOS to model the range of decision-making functions that would be made by an autonomous operator. The complete set of basic actions consists of the following:

- Reaching and grasping a device with a hand or foot.*

*At present, HOS models body movements at the level of individual hands, feet, eyes; movements of fingers, elbows, or knees are not modeled. Thus, some difficulty would be encountered if the present version of HOS were used to simulate the fine-grained details of the operation of a typewriter keyboard. However, these limitations are not expected to be significant in any of the currently anticipated applications for HOS.

- Looking at a device -- moving the visual fixation point to the device location.
- Absorbing information from a display, control, or symbol via vision or touch.
- Manipulating a control.
- Attempting to recall the value of a device or function.
- Performing the computation of an operator function.

A.4 THE MULTIPLEXING MODEL

As stated previously, the operator, (1) can place procedures on an active procedure list, (2) may or may not execute them immediately, and (3) can remove them when completed or no longer needed. The multiplexing model controls this selection of procedures -- it selects the procedure to be executed whenever a procedure is completed or interrupted. While the HOS operator can execute only one procedure at a time, individual actions can occur simultaneously if they do not require the same physical resources; virtual parallel processing can be achieved through rapid switching between procedures. Procedural selection takes into account the type of procedures on the active procedure list (regular, enable, adjust, or disable); the initial criticality for each procedure as supplied by the analyst; the amount of time elapsed since the procedure was last executed; and, for procedures that monitor continuous devices, how close the value of the device is to its desired value. The multiplexing algorithm has been designed to give an intuitively reasonable weight to each of these factors.

There are several principles according to which the multiplexor operates; they are:

(1) If all procedures on the active procedure list are interrupted, then the simulation time is incremented until one of the procedures is no longer interrupted.

(2) If there is an uninterrupted enable procedure on the active procedure list, then that procedure is given absolute priority over all other procedures.

(3) If steps 1 and 2 have not selected a procedure, then an effective criticality is computed for each uninterrupted procedure and the procedure with the highest effective criticality is selected. The effective criticality, ECRIT, is defined as:

$$\text{ECRIT} = \text{CRIT} * \text{TMULT} * \text{ADMULT}$$

CRIT -- is the initial criticality supplied by the analyst.

TMULT -- is a function of the time elapsed since the last execution of the procedure.

ADMULT -- is a factor that, for monitor procedures, measures how close the estimated value of the monitored device is to its desired value.

(4) The function currently used for TMULT is:

$$\text{TMULT} = 1 + \log_{10}(1 + t)$$

where t is elapsed time since the last execution. ADMULT equals 1 for all procedures other than monitor procedures, for which:

$$\text{ADMULT} = 1 + \left|\frac{\text{E - DESIRE}}{\text{UPPER - DESIRE}}\right| .$$

where

E -- is the operator's estimate for the value of the device linearly extrapolated to the present time.

DESIRE -- is the desired value of the device.

UPPER -- is the upper limit for the desired value of the device.

Top priority is given to enable procedures since they are always prerequisite to the execution of some other procedure and generally consume very little time. Relative priorities for other procedures can be established by the analyst through the choices of initial criticalities for the procedures. Criticalities can also be reassigned by the analyst throughout the simulation.

The functions for TMULT and ADMULT were chosen so that neither factor would completely dominate the other. Since adjust procedures are expected to keep the estimated value of the adjusted device within the prescribed limits, ADMULT is expected to vary between 1 and 2. Clearly, when ADMULT is near 2 (i.e., when the estimated value of the device is near one of its limits), the adjust procedure should have a high criticality. Accordingly, TMULT was defined so that its range would be comparable to that of ADMULT over the domain of elapsed times appropriate to HOS procedures. Thus, TMULT = 2 when the elapsed time is 9 seconds and TMULT = 3 when the elapsed time is 99 seconds.

In choosing initial criticalities for procedures or in modifying criticalities during a simulation, careful attention must be given to the interaction between the initial criticality and TMULT. Ignoring, for the moment, the contribution of ADMULT, the initial criticality of a procedure can be interpreted as a factor that determines how long a procedure must wait on the active procedure list before it attains the same effective criticality as another procedure which has either just been added to the list or just completed a portion of its execution. For example, if procedures A and B are on the list and have initial criticalities of C_A and C_B, respectively, with $C_A > C_B$, the procedure A will be executed first. Whenever procedure A is completed or interrupted, $TMULT_A$ will be exactly 1, since the elapsed time since execution of that procedure is zero. $TMULT_B$, on the other hand, will be greater than 1. As long as procedure A remains on the list, procedure B will have to wait until $TMULT_B \cdot C_B > TMULT_A \cdot C_A$ before it will be executed. Since $TMULT_A$ will always be exactly 1, the minimal waiting time for procedure B will be the time at which $TMULT_B = C_A/C_B$. Table A-2 indicates the minimal waiting times for procedure B for a variety of values of the ratio C_A/C_B. Note that when there are more than two procedures on the list, the comparison standard (procedure A) should always be the procedure with the highest initial criticality since that procedure will dominate.

Table A-2. Minimal Waiting Times as a Function of the Criticalities of Procedures A and B

RATIO C_A/C_B	WAITING TIME (SECS.)
1.000	0
1.176	.5
1.301	1.0
1.477	2.0
1.602	3.0
1.699	4.0
1.778	5.0
1.845	6.0
1.903	7.0
1.954	8.0
2.000	9.0
2.041	10.0
2.415	25.0
2.708	50.0
3.005	100.0

A.5 THE ANATOMY MOVEMENT MODELS

The HOS anatomy movement model determines which body part will perform each specific perceptual and manipulative task and how much time will be consumed by the movement required to accomplish each task. Also, to maintain versimilitude, the anatomy movement model moves a body part to a "relaxed" location whenever the body part is to remain inactive for a period of time specifiable by the analyst. The model currently describes eye, hand, and foot movements. Each hand and foot movement time is determined according to a straight line path between the current location and the desired location, whether or not the crewstation geometry would actually permit such a movement. All movements are ballistic, i.e., they cannot be altered or terminated mid-course. The hand and foot movement precision required varies with the dimensions of the device to which they are being moved and is thus included in the movement time calculation, thereby compensating for any fine positioning movements that may be necessary.

A.5.1 Body Part Selection

The determination of the body part to be used in accomplishing any task is made according to a set of common-sense principles in conjunction with the constraints that the analyst places on which body parts can read and manipulate each display and control. Devices can be characterized such that:

- Only the eyes can read the device; manipulations are not possible.
- The eyes are preferred for reading, but the hand may also be used. Hands are used to manipulate, but the simulation will require the operator to look at the device before moving his hand to it.
- The hands are used for both reading and manipulating and the operator will not look at the device before moving his hand to it.
- The feet are used both to read and to manipulate the device.

Body movements are initiated by either an explicit instruction to read or manipulate a device or by an instruction to look at or grasp a device. The logic that determines which body part will be used is somewhat different for these two situations. The GRASP/LOOK AT instruction enables the analyst to direct the operator to move a body part to a device in anticipation of the fact that he will later have to read or manipulate the device. Consequently, the location of the device becomes the "grasp location," i.e., the location at which the assigned body part is to be kept when not needed elsewhere, until some action (reading or manipulating) is performed by that body part at that location or until another "grasp location" is established for the body part. If the body part is moved away from its "grasp location" in order to read or manipulate another device, it will be moved back to the "grasp location" as soon as the other action is completed.

When the instruction is to "look at" a device, there is, of course, no difficulty in determining the appropriate body part for the task. However, when the device is to be grasped by a hand, HOS must decide whether to use the right or left hand. Figure A-1 illustrates the logic used in making that decision. If the hand closest to the device has no assigned grasp location, then that hand is used. Otherwise, HOS attempts to maintain any old grasp locations, if possible, by using the less preferred hand or by switching task assignments for the two hands. Grasp locations for the feet are determined according to exactly the same logic as the hands with the exception that the foot assignments will not be swapped to maintain an old grasp location.

The logic for determining which hand will perform a reading or manipulating task is illustrated in Figure A-2. The variable TINC is a user-specified time value which represents the maximum amount of time that the simulated operator will wait for an occupied body part to become available before he will attempt to find another means for completing a task. If the hand closest to the device of interest is available now or will be available within TINC seconds, then that hand will be used for the action.

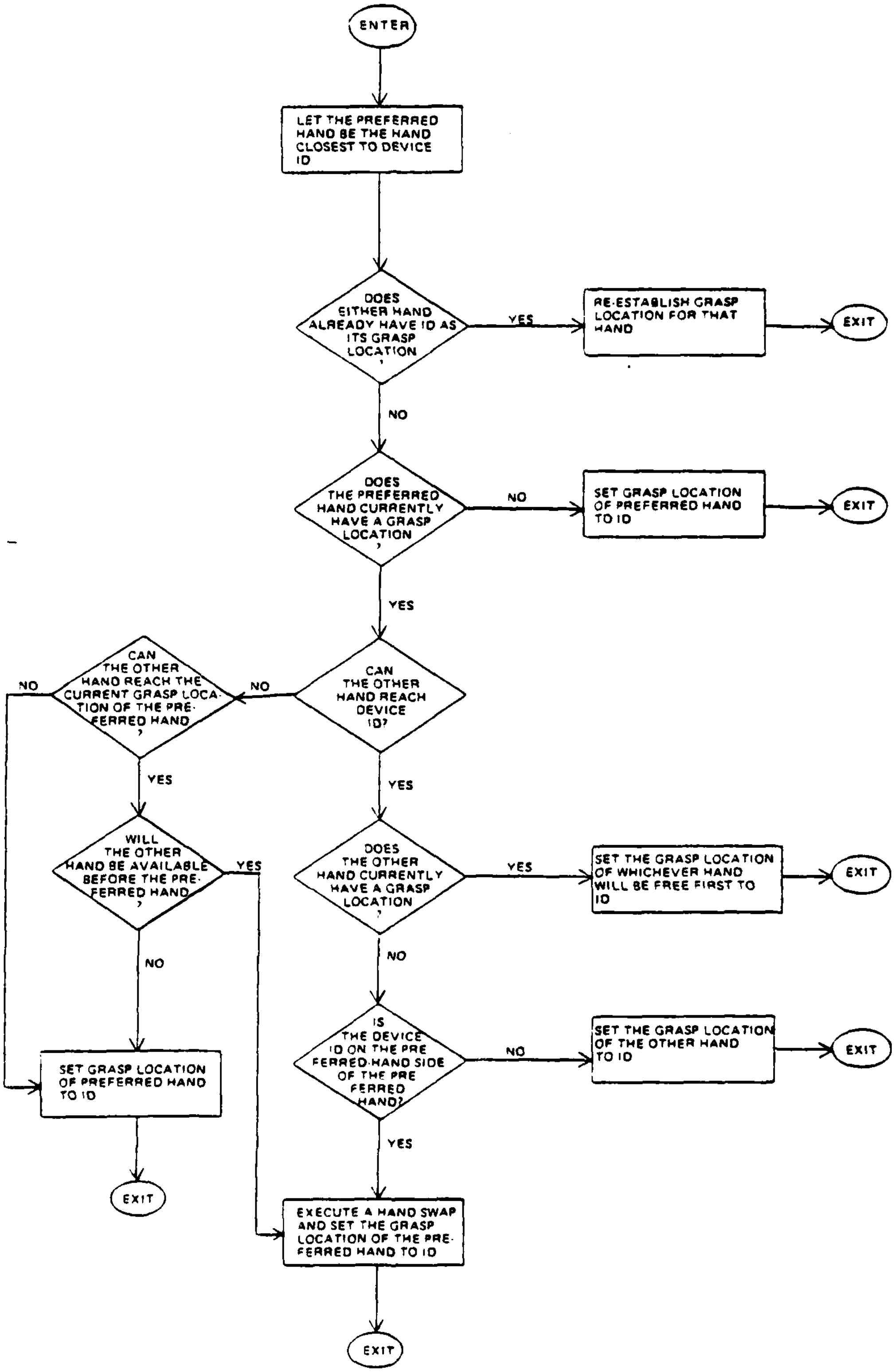

ENTER
LET THE PREFERRED HAND BE THE HAND CLOSEST TO DEVICE ID
DOES EITHER HAND ALREADY HAVE ID AS ITS GRASP LOCATION ?
YES
RE-ESTABLISH GRASP LOCATION FOR THAT HAND
EXIT
NO
DOES THE PREFERRED HAND CURRENTLY HAVE A GRASP LOCATION ?
NO
SET GRASP LOCATION OF PREFERRED HAND TO ID
EXIT
YES
CAN THE OTHER HAND REACH DEVICE ID?
NO
CAN THE OTHER HAND REACH THE CURRENT GRASP LOCATION OF THE PREFERRED HAND ?
NO
YES
WILL THE OTHER HAND BE AVAILABLE BEFORE THE PREFERRED HAND ?
YES
NO
SET GRASP LOCATION OF PREFERRED HAND TO ID
EXIT
YES
DOES THE OTHER HAND CURRENTLY HAVE A GRASP LOCATION ?
YES
SET THE GRASP LOCATION OF WHICHEVER HAND WILL BE FREE FIRST TO ID
EXIT
NO
IS THE DEVICE ID ON THE PREFERRED-HAND SIDE OF THE PREFERRED HAND?
NO
SET THE GRASP LOCATION OF THE OTHER HAND TO ID
EXIT
YES
EXECUTE A HAND SWAP AND SET THE GRASP LOCATION OF THE PREFERRED HAND TO ID
EXIT

Figure A-1. Logic of Grasp Location Assignments

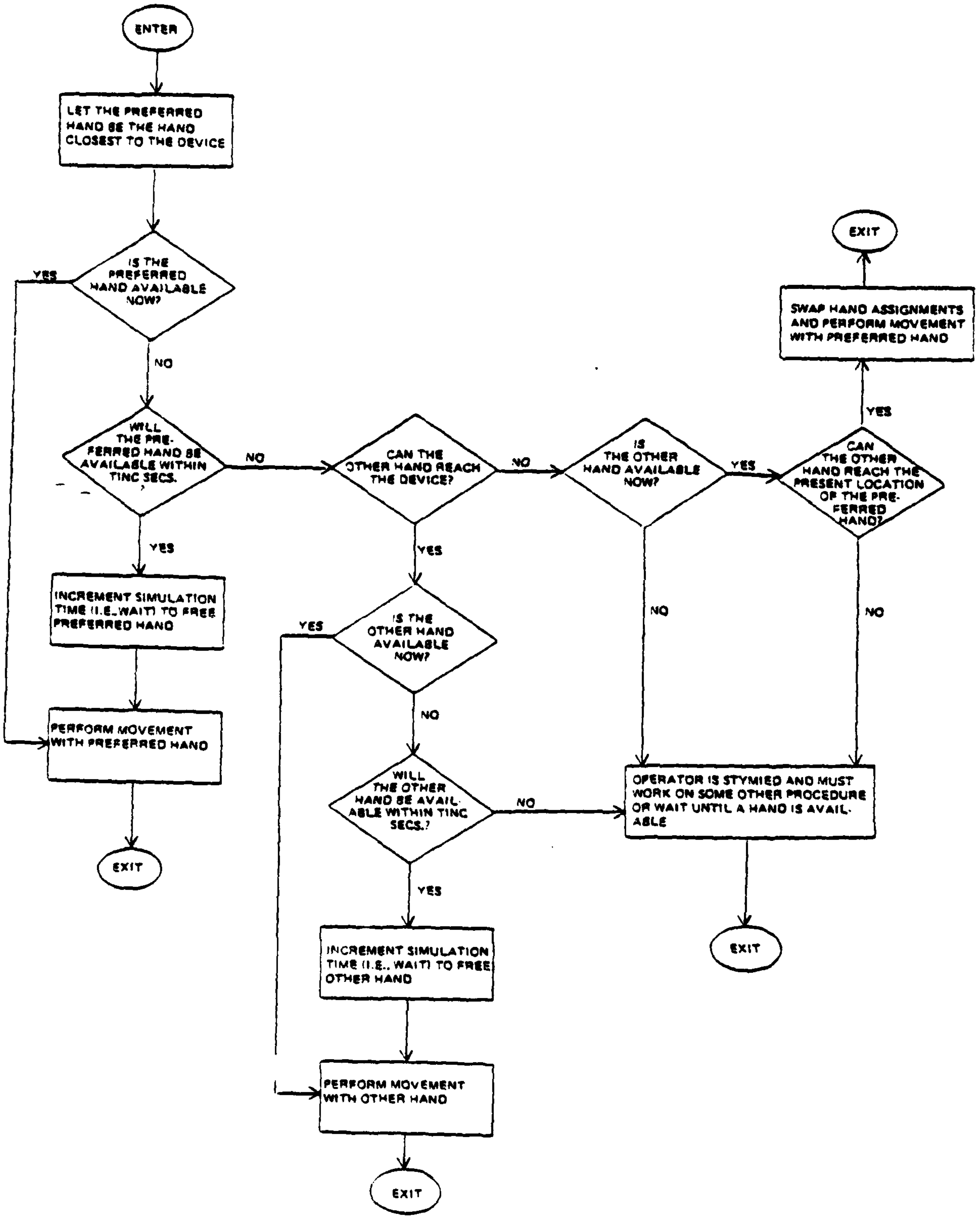

ENTER
LET THE PREFERRED HAND BE THE HAND CLOSEST TO THE DEVICE
IS THE PREFERRED HAND AVAILABLE NOW?
YES
NO
WILL THE PRE-FERRED HAND BE AVAILABLE WITHIN TINC SECS.?
NO
YES
INCREMENT SIMULATION TIME (I.E. WAIT) TO FREE PREFERRED HAND
PERFORM MOVEMENT WITH PREFERRED HAND
EXIT
CAN THE OTHER HAND REACH THE DEVICE?
NO
YES
IS THE OTHER HAND AVAILABLE NOW?
YES
NO
WILL THE OTHER HAND BE AVAIL-ABLE WITHIN TINC SECS.?
NO
YES
INCREMENT SIMULATION TIME (I.E., WAIT) TO FREE OTHER HAND
PERFORM MOVEMENT WITH OTHER HAND
EXIT
IS THE OTHER HAND AVAILABLE NOW?
YES
NO
CAN THE OTHER HAND REACH THE PRESENT LOCATION OF THE PRE-FERRED HAND?
YES
NO
SWAP HAND ASSIGNMENTS AND PERFORM MOVEMENT WITH PREFERRED HAND
EXIT
OPERATOR IS STYMIED AND MUST WORK ON SOME OTHER PROCEDURE OR WAIT UNTIL A HAND IS AVAIL-ABLE
EXIT

Figure A-2. Logic of Reading/Manipulating Body Part Assignments

Otherwise, the other hand will be used if it can reach the device and is available. If the other hand cannot reach the device but can take over the task which the preferred (closest) hand is performing, then the hand assignments are swapped. If none of these conditions is satisfied, then the operator is stymied and unable to continue work on the current procedure until the situation changes.

The logic for determining the assignment of feet for reading and manipulating is exactly the same as for hands, except that foot swapping is not allowed. For devices that can be read by either the eyes or the hands, the eyes will always be used unless a hand is already in contact with the device.

A.5.2 Movement-Time Models

The times associated with each body movement are based on formulas derived from a variety of human performance studies. These equations predict movement times that are based solely on the magnitude of a movement; thus, there is no variability as is observed in actual human performance -- the current equations do not attempt to model individual differences or random variations in movement times. Thus, at present, the equations describe a perfectly consistent average operator.

Eye movement times are determined as a function of the changes in angles of focus and onvergence, as shown in Figure A-3. The change in the angle focus, as shown in Figure A-3a, is the angle between the two fixation points and the design eye reference point. If the vectors from the design eye reference point to the two fixation points are P_1 and P_2, then by the law of cosines, the change in angle of focus will be:

$$\Delta\theta = \cos^{-1}\left(\frac{P_1 \cdot P_2}{|P_1|\ |P_2|}\right)$$

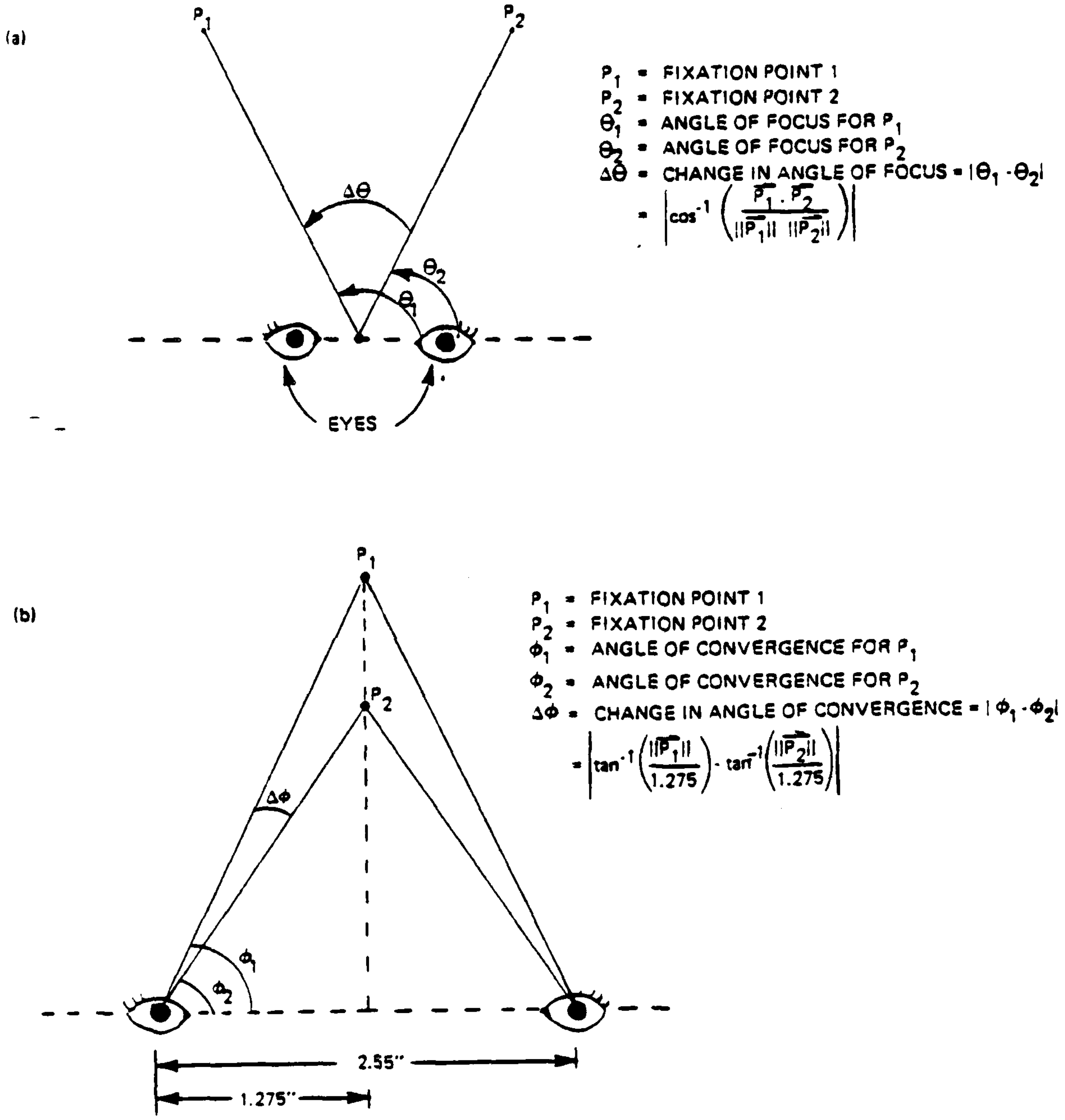

(a)
P_1
P_2
P_1 = FIXATION POINT 1
P_2 = FIXATION POINT 2
θ_1 = ANGLE OF FOCUS FOR P_1
θ_2 = ANGLE OF FOCUS FOR P_2
$\Delta\theta$ = CHANGE IN ANGLE OF FOCUS = $|\theta_1 - \theta_2|$
$= \left|\cos^{-1}\left(\frac{\overrightarrow{P_1}\cdot\overrightarrow{P_2}}{\|\overrightarrow{P_1}\|\ \|\overrightarrow{P_2}\|}\right)\right|$
$\Delta\theta$
θ_2
θ_1
EYES
(b)
P_1
P_2
P_1 = FIXATION POINT 1
P_2 = FIXATION POINT 2
ϕ_1 = ANGLE OF CONVERGENCE FOR P_1
ϕ_2 = ANGLE OF CONVERGENCE FOR P_2
$\Delta\phi$ = CHANGE IN ANGLE OF CONVERGENCE = $|\phi_1 - \phi_2|$
$= \left|\tan^{-1}\left(\frac{\|\overrightarrow{P_1}\|}{1.275}\right) - \tan^{-1}\left(\frac{\|\overrightarrow{P_2}\|}{1.275}\right)\right|$
$\Delta\phi$
ϕ_1
ϕ_2
2.55"
1.275"

Figure A-3. Derivation of Eye Movement Equations where $P_1 \cdot P_2$ denotes the dot product of P_1 and P_2, and $|P_1|$ and $|P_2|$ are the magnitudes of P_1 and P_2, respectively. The change in the angle of convergence, as shown in Figure A-3b, is calcualted by assuming that fixation points are in the sagittal plane of the operator. Thus, the angle of convergence is the angle that the line of sight for either eye makes with the line connecting the two eyes. For an interpupillary distance of 2.55 inches (corresponding to a 50th percentile USAF pilot), the change in the angle of convergence, $\Delta\phi$, will be:

$$\Delta\phi = \left| \tan^{-1}\left(\frac{P_1}{1.275}\right) - \tan^{-1}\left(\frac{P_2}{1.275}\right) \right|$$

Eye movement angles, $\Delta\theta$ and $\Delta\phi$, when related to movement times in an experiment involving lateral eye movements (conducted by Dodge and Cline, 1901) and an unpublished study that involved both lateral and convergence movements (R.J. Wherry and A. Bittner), yield the formula:

$$T = .14324A + .0175$$

where $A = \max(\Delta\theta,\Delta\phi) + .2 \min(\Delta\theta,\Delta\phi)$ in which $\Delta\theta$ and $\Delta\phi$ are expressed in radians and T, the movement time, is in seconds.

It should be noted that this model, while accurately representing the experimental results, describes only ballistic eye movements and does not consider the processes of visual search, accommodation, adaptation, or interpretation.

The equations for hand and foot movement times are derived from adaptations of Fitts' model for speed and accuracy of body movements (Fitts, 1954; Fitts and Peterson, 1964). Fitts' law, derived from Shannon and Weaver's theory of information transmission (Shannon and Weaver, 1949), states that movement time is a linear function of the "information content" of a movement. The information measure, which Fitts called an index of difficulty (ID) is defined as:

$$\text{ID} = \log_2 \frac{2A}{W}$$

where A is the amplitude of the movement and W is the permissible range of terminal movement error or target bandwidth. The validity of this law has been demonstrated for a variety of self-paced, repetitive movements between two targets (Fitts, 1954) and for discrete movements under a variety of uncertainty conditions (Fitts and Peterson, 1964). Other investigators have suggested a modification of the definition of ID which provides a slight improvement in the descriptive accuracy of the law (Welford, 1960; Drury, 1975). The revised index of difficulty (ID^1), which was designed chiefly to predict movement times near zero for ID values near zero, is defined as:

$$\text{ID}^1 = \log_2 \left(\frac{A}{W} + .5 \right)$$

It has also been suggested that the model parameter for control size, W, be interpreted as an empirical measure of movement accuracy rather than as a directly observable dimension of a device (Crossman, 1960; Welford, 1960; Drury, 1975). Thus, the value for W should reflect the dimensions of the hand or foot that operates a control and the manner in which the control is contacted (thumb and index finger grasp, index finger depression, etc.) as well as the size of the device.

Despite the widespread acceptance in the human performance literature of some version of Fitts' law, a re-analysis of the data shows that a superior funcitonal description is possible. Fitts' law implies that movement-time can be decomposed into two components, one of which is a function of the amplitude of the movement, the other a function of the accuracy of the movement (i.e., the width of the target), and that the total movement time is just the sum of these two component times.

It can be shown that the best decomposition of the total movement times into two additive components (i.e., the decomposition that minimizes summed squared error) consists of assigning the marginal means plus an arbitrary constant to the level of one of the components and then assigning to each level of the other component the mean (for that level) of the remaining movement times. Thus, it is possible, objectively, to separate the total movement times obtained by Fitts into two components, which we will call travel time and aiming time, without making any assumptions other than that the components are additive. The decomposition is unique only up to an arbitrary additive constant, although the restriction that the component times must be positive severely limits the allowable range for the constant. In any case, the shapes of the empirical functions relating travel time to distance moved and aiming time to target size are uniquely determined by the data. We have decomposed Fitts' movement time data in this manner to obtain travel time and aiming time components and then examined each of the components separately to determine what mathematical function would best relate the component time to the distance moved or target size. This stragegy avoids the assumption inherent in Fitts' approach, that both components must be described by particular logarithmic functions. In fact, a logarithmic function does provide a very good fit to the obtained aiming time data for which we found

$$t_A = .135 - .096 \log_2 W$$

where t_A is the aiming time and W is the target width. The travel time data, however, appears to be much too linear to permit an acceptable fit by a logarithmic function. The travel time function is clearly negatively accelerated, so even though a straight line fits the data better than any logarithmic function, the linear function is also unacceptable. Figures A-4a and A-4b display the empirical functions for travel time and aiming time along with the best fitting logarithmic function for Fitts' one ounce stylus data.

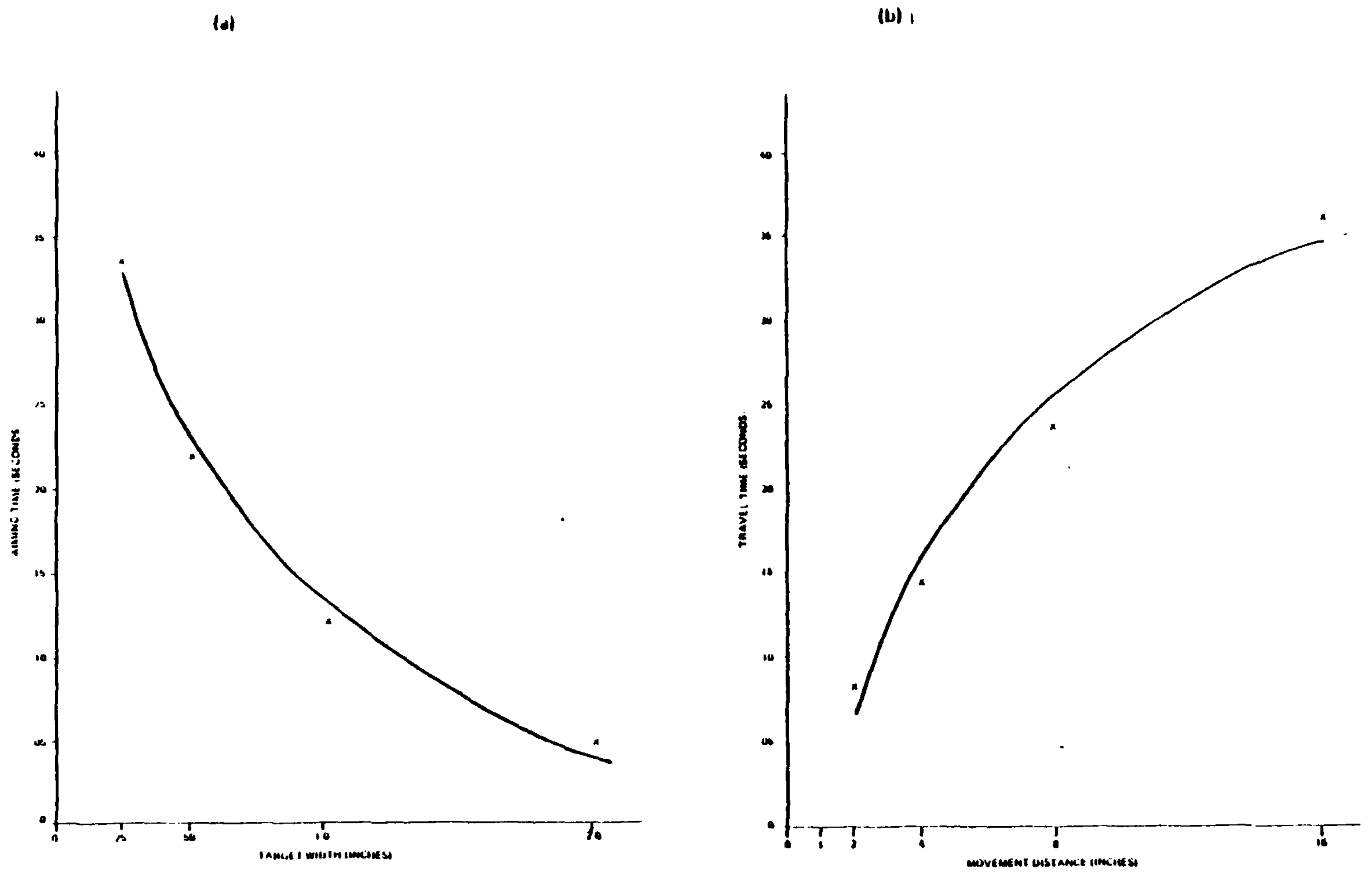


Figure A-4. Decomposed Travel and Aiming Time Functions for Fitts' One Ounce Stylus Reciprocal Tapping Data

Although HOS requires a model for discrete movements, the previous analysis was based on data obtained for repetitive movements because of the lack of available appropriate data for discrete movements. Although Fitts and Peterson (1964) performed the appropriate experiments with discrete movements, they report their movement time data only as a function of ID value rather than reporting separate times for each combination of target size and movement distance. The best linear fit between ID value and movement time for their data is:

$$MT = .074\ ID - .070$$

where MT is total movement time in seconds. Since Fitts and Peterson report the same pattern of results for discrete movements, as was previously obtained for repetitive movements, it seems reasonable to assume that the aiming time components of this function would be acceptable, while the travel time component would be better described by some other function. Thus, the model for aiming time currently used in HOS is:

$$t_A = \begin{cases} 0 \text{ for } w > 2 \\ .074 - .074 \log_2 W \text{ for } 0 < w \leq 2 \end{cases}$$

where the additive constant was chosen to predict zero aiming time for a two-inch target. The cutoff of two inches for target width was employed because the available data did not include any larger targets and because the aiming time curve appeared fairly level at that point. We have examined a variety of functional descriptions for travel time, but the available data has not permitted a definite choice to be made. Although HOS currently employs a travel time model (suggested by Topmiller and Sharp, 1965), we anticipate modifying this model in the near future. The relationship proposed by Topmiller and Sharp is:

$$t_T = \frac{d}{1.75d + 8.99}$$

where t_T is travel time and d is distance in inches. This model is favored over Fitts' model for travel times because it was derived by fitting a curve to movement times obtained for movements to contro' switches on a simulated control panel. This situation is deemed much more relevant for HOS applications than are the movements of a stylus to rectangular targets, as studied by Fitts and Peterson.

It has recently been demonstrated (Drury, 1975) that Fitts' law applies quite well for repetitive foot movements between coplanar pedals. For this situation, Drury found that the best linear function for relating MT to ID^1 was:

$$MT_{repetitive} = .187 + .085\ ID^1$$

where ID^1 is as defined previously and the effective pedal width, W, is defined as the sum of the physical pedal width and the operator's shoe width. This formula can be converted to a formula for discrete movements, Drury argues, by multiplying by the factor relating repetitive hand movement times to discrete hand movement times for the Fitts studies. (He reports that factor as $MT_{discrete}/MT_{repetitive} = .61$). Drury's formula for discrete movements between coplanar pedals is, therefore:

$$MT_{discrete} = .114 + .053\ ID^1$$

Limited experiments by Davies and Watts (1969), in which discrete movements between coplanar pedals at a single separation were compared with movements between a single pair of non-coplanar pedals suggest that this formula can ge generalized to non-coplanar pedals. Their results showed that it took an average of .149 seconds to move 6.5 inches between coplanar pedals and .309 seconds to move between the pedals when one pedal was raised 6 inches. Assuming that the time penalty for non-coplanar movements over coplanar movements is proportional to the ratio of the required change in leg extension to the amplitude of the movement, movement time would be predicted by the general formula:

$$MT = \left(1.0 + 1.4\ \frac{\Delta E}{A}\right)\left(.114 + .052\ ID^{1}\right) \text{ seconds}$$

where ΔE is the change in leg extension required by the movement. We expect that this model will be modified as additional relevant data become available. For the present, it accurately predicts foot movement times between pedals of common dimensions at some common separations consonant with the data of Davies and Watts (1969) and Drury (1975).

Times associated with manipulating controls are largely left to the user since general formulas that predict manipulation times for the diverse types of contemporary controls are not available. For discrete controls, HOS requires that the user specify the time that it takes the operator to move the control between two adjacent settings. The time consumed by a manipulation of more than one setting is then calculated as the product of the number of settings to be moved through and the input time cost for a single setting cha ge. In assigning the input time costs for discrete devices, it is thus incumbent on the user to determine the most salient factors that characterize each control to be modeled and then to consult a reliable source in order to obtain empirical manipulation time data for that type of control. For general information on the importance of the various control design factors, we refer the reader to Chapanis and Kinkade (1972). For more detailed information about the operation of keyset devices, we suggest that the reader consult Seibel (1972). A variety of sources offer experimental data, obtained by several different methods, on the times for operating discrete devices -- Bradley and Wallis (1959, 1960), Dean, Farrel, and Hitt (1969), Goldbeck and Charlet (1974), and MacPherson and Siegel (1967).

An explicit formula for the operation of continuous rotary controls is included in HOS, requiring only that the user specify the force needed to turn the dial. This formula, derived by fitting a simple quadratic function to a table of idealized data (presented by Karger and Bayha, 1966), is:

$$T = .0482 + .0050F + .0825A + .0084\ FA$$

where F is the force in pounds needed to turn the dial and A is the angle through which the dial is to be turned in radians. Unfortunately, we have not found any data that can be used to assess the validity of this formula. However, through the study of video-tape records of control operations in the course of a protracted mission, Goldbeck and Charlet (1974) determined that the mean time that the operator's hand stayed in contact with any continuous rotary control on a single manipulation was .73 seconds. Of course, this time may include some idle dwell time as well as multiple manipulations that were not discriminated during data analysis. Assuming that the forces required to turn the knobs in that study were about one pound and the magnitudes of the turns were of the order of 60^o to 120^o (roughly one to two radians), then the HOS model would predict manipulation times in the vicinity of .2 seconds. This comparison suggests that HOS may underestimate control manipulation times, but revising the model would not be appropriate until sufficiently comprehensive data (including complete physical specifications of the controls) becomes available.

A.6 THE COGNITIVE MODELS

The information processing functions modeled in HOS include information absorption, memory, and mental computation. The models for these three operations are intricately interrelated through a variable termed hab strength that is associated with each device or mental calculation and represents the durability or strength of the operator's knowledge of its value. The models are also related to one another by informational task demands as depicted in the flowchart of Figure A-5. The information processing models may be accessed either directly by instruction (as indicated by the double arrow at the top of the figure) or indirectly by the requirements of another process, such as when the computation of an operator function (the HOS term for mental calculation) depends on the absorption (the HOS term for perception) of the value of a display or control.

In examining the HOS cognitive models, it is appropriate to analyze the implications of each of the following considerations for each model:

(1) Type of information to be handled.

(2) Aspects of performance to be described.

(3) Specifications of processing details.

(4) Interpretation and estimation of model parameters.

(5) Experimental situations appropriate for model verification.

Each point, except the first, will be treated separately for each of the three models with which we are concerned. Since the information to be handled by the models is essentially the same for all, the first point will be discussed only once. Also, since the hab strength concept is basic to all three models, we will present a description of how hab is used and modified by the models before we discuss the specific models.

A.6.1 Types of Information

Device values are represented in HOS as discrete settings, real numbers, and ordered pairs of real numbers for devices that are declared as discrete, continuous, and positional, respectively. Each such value is treated as a single item or "chunk" although the range of possible values may vary considerably across devices. Values of operator functions are less restricted in principle than are device values. However, function calculations will generally represent operations performed on device values and results of function calculations will frequently be compared with device values; thus, function values can also be assumed to represent either discrete settings, real values on continuous scales, or positional quantities (ordered pairs of real numbers).

HOS actually maintains several parameter values that relate to the operator's knowledge of a device or operator function value, although only one represents the operator's conscious knowledge of the current value. In particular, the information stored by HOS includes:

- The actual value of a device or function.
- The operator's current estimate for the value of a device or function, the time when that estimate was obtained, and the hab strength associated with that estimate.
- The estimated value that the operator obtained immediately prior to his current estimate, the time when that prior estimate was obtained, and its associated hab strength.
- The desired value for a device or function.
- The desired upper and lower limits for continuous devices.

Of these values, only the current estimated value is obtained directly by the absorption, recall, and mental computation models. The others (except for the desired value and desired limits which are not used by the cognitive models) are used only in defining the functional forms of the models and do not represent values of which the operator is consciously aware.

All information that is absorbed, recalled, or computed by the simulated operator is limited to the categories of discrete settings, real numbers, and ordered pairs of real numbers. The operator does not have to recall the steps in the procedures, and differences between two displayed values are not perceived directly through the absorption model. However, the HOS user has considerable freedom in defining what is to be considered a display or control, and operator functions may be constructed that perform functions other than simple numerical calculations. For example, a clock wth an hour hand and a minute hand may be treated as a single display for time or it may be treated as two distinct displays sharing the same scale, one indicating hours and the other minutes. These two ways of describing the display might actually correspond to two different display reading strategies available to the operator -- alternate ways of "chunking" information. The consideration of how information is "chunked" will be relevant to the evaluation of all three cognitive models.

A.6.2 Hab Strength Modification

The concept of hab strength underlies all of the HOS models for cognitive functions. It determines whether or not recall attempts will succeed and how much time will be consumed by the processes of recall, perception, and mental calculation. Like its namesake in the elaborate stimulus-response (S-R) learning theory of Clark Hull, it represents a measure of how well something is learned. However, unlike the Hullian concept of hab strength, which indicates the relative intensity with which a particular stimulus tends to evoke a particular response, the HOS concept of hab strength indexes the durability of an item of information in a temporary memory store.

The hab strength for an item is modified when an estimated value is obtained by absorption or computation or when the value is successfully recalled from memory; the same formula is used to compute a new hab strength whenever any of these events occur. The hab strength is modified once upon the completion of the recall process; for the absorption and computation processes, it is modified once for each "micro-absorption" process or "micro-computation." The new hab strength is observed from the old by the formula:

$$H_1 = V + (1-V) \cdot H_0 \cdot \left(\frac{S}{1 + (t-R)\ k} \right) \qquad (1)$$

where

H_1 -- is the new hab strength.

H_0 -- is the old hab strength.

R -- is an input constant (REMEM) representing the memory. cycle time

k -- is an input constant (HABFAC) used as a scale factor.

t -- is the time since the estimated value was last determined.

S -- is a measure of the similarity between the current and previous estimated values of an item.

V -- is a base level for the hab strength of that item.

The measure of similarity between two estimated values, S, ranges between 0.1 and 1.0. For discrete devices, S is always 1.0. For continuous devices and functions, S is defined by the formula:

$$S = \begin{cases} 1.0 & \text{if } A \leq 1 \cdot \\ e^{-[(A-1)/6]^2} & \text{if } 1 < A < 10.105 \\ 0.1 & \text{if } A \geq 10.105 \end{cases}$$

where

$A = \left|\frac{E-PE}{TOL}\right|$

E = current estimated value.

PE = previous estimated value.

TOL = user-supplied parameter representing a desired accuracy tolerance for each device.

The minimum value of the hab strength for a device or function, V, is a constant between 0.1 and 1.0 for that device or function. V may be interpreted as the minimum hab strength that would be assigned if S were zero (though S can actually never be smaller than 0.1). Values for V are dependent upon the declared character of each device and function as follows:

$V = 0.1$	for discrete devices with more than six possible settings and all functions and continuous or positional devices.
$V = \frac{1}{n} - .05$	for all discrete devices for which the number of possible settings, n, is six or less.
$V = .95$	for all momentary devices

These values can be loosely interpreted as the probabilities of "guessing" the value of a device, i.e., the minimum hab strength is higher for devices with lower inherent uncertainty.

Two constraints are imposed on Equation 1 in its use in HOS in order to maintain hab strength in the interval between 0.1 and 1.0. First, if H_0 is less than or equal to 0.1, then H_0 is set to 0.1 and t is set to zero. Secondly, if application of Equation 1 produces a value of H_1 greater than 1.0, then H_1 is set equal to 1.0.

In discussing the implications of Equation 1, it will be useful to rewrite it as:

$$H_1 = V + (1-V) \cdot H_0 \cdot M \qquad (2)$$

where

$$M = \frac{S}{1 + (t-R)\ k}$$

The factor M then represents the influence of the previous determination of an estimated value on the new hab strength. In order to ensure that M is positive and finite, k must be chosen so that $\frac{1}{R} > k \geq 0$. That is, k must be chosen to be non-negative and smaller than the reciprocal of R. Then, M will be larger than S whenever t is less than R, and M will be smaller than S when t exceeds R.

In order to understand the HOS models for absorption and function computation, it will be useful to describe the effects of repeated applications of Equation 2. Such a description is only feasible if M reamins constant for successive applications, i.e., if S is constant and either t is constant or k is zero. In many interesting siutations in HOS, the factor M does remain constant for successive applications of Equation 2. If H_j denotes the hab strength after the j^{th} application of Equation 2 when the starting hab strength was H_0, then it can be shown that:

$$H_j = h - \left(1-V\right)^j M^j \left(h - H_0\right) \quad \text{if } M < \frac{1}{1-V} \qquad (3)$$

where

$$h = \frac{V}{1 - (1-V)\ M}$$

Note, that if j increases without bound, the H_j will approach h. That is,

$$H_\infty = \lim_{j \to \infty} H_j = h \text{ for } M < \frac{1}{1-V} \qquad (4)$$

H_∞ is thus the asymptotic value of successive applications of Equation 2 when M is not too large. We can use Equation 4 to rewrite Equation 3 as:

$$H_j = H_\infty - \left(1-V\right)^j M^j \left(H_\infty - H_0\right) \quad \text{for } M < \frac{1}{1-V} \qquad (5)$$

which incicates how each successive application of Equation 2 brings the hab strength closer to H_∞.

In addition to knowing the asymptotic value for hab strength, it will be important to know how rapidly this value is approached and, in particular, how many applications of Equation 2 will be required in order for hab strength to exceed a predetermined value. This is important because in the HOS models for absorption and computation, micro-attempts at absorption or computation are repeated with the hab strength being incremented according to Equation 2 on every micro-attempt until hab strength exceeds a user-supplied value. This value represents the certainty threshold that the operator must attain before he is "satisfied" with the result. We will discuss the consequences of this model under the assumption that M is constant in Equation 2, by using Equation 3. In particular, we will want to know how many micro-attempts will be required to raise a hab strength, H_0, less than the threshold to a value that exceeds the threshold. For this to be a meaningful problem, we must require that the asymptotic value for hab strength, H_∞, be greater than the threshold since, otherwise, the hab strength will never exceed the threshold. Letting L be the threshold and H_k the hab strength after the k^{th} micro-attempt, we can convert Equation 5 to a continuous function of j and apply a logarithmic transformation to determine the number, N_L, for which $H_{N_L} < L \le H_{N_L}$. The solution is:

$$N_L = I_g \left[\frac{\ln\left(H_\infty - L\right) - \ln\left(H_\infty - H_0\right)}{\ln\left[\left(1-V\right) M\right]} \right] \qquad (6)$$

where $I_g(X)$ is the "greater integer" function, i.e., the smallest integer that is greater than or equal to X.

A second constraint placed on the growth rate of H_k is based on the idea that if the increment in hab strength resulting from a micro-attempt is too small, then the operator should stop trying to absorb or compute a value. As in the preceding case, we are interested in knowing how many micro-attempts can be allowed before the absorption or computation process will be stopped by this limit, d. Again, if we assume that the same value of M applies for all micro-attempts, then the absorption process will be terminated at the $N_d{}^{th}$ iteration where:

$$N_d = I_g \left[\frac{\ln d - \ln \left[1 - (1-V)\,M\right] - \ln \left(|H_\infty - H_0|\right)}{\ln \left[(1-V)\,M\right]} + 1 \right] \quad (7)$$

Under the stated assumptions, the preceding derivations for H_∞, N_L, and N_d represent precise determinations. The most severe restriction was that M in Equation 2 had to be constant for successive micro-attempts. Although this assumption is valid for many absorption and function computations, it is also desirable to know something about H_∞, N_L, and N_d when this is not the case. Since M depends on both the similarity between successive estimated values and the time interval separating their determinations, if variations in similarity and time separation between estimates for an item are reasonably erratic, then M can be treated as a random variable. If this is the case, M can be replaced by its expected value in Equations 4, 6, and 7, and the formulas will represent expected operator approximations for H_∞, N_L, and N_d, respectively.* However, if M varies systematically across

*See Bush and Mosteller (1956) for a discussion of expected operator approximations for learning models similar to Equation 2.

successive applications of Equation 2, then the values for H_∞, N_L, and N_d must be derived in accordance with the specific manner in which M varies.

A.6.3 The Function Computation Model

Operator functions enable HOS to model the processes by which the operator derives values from directly observed information. Thus, they represent the arithmetic and logical functions that are carried out "in the operator's head." In addition, operator functions enable the analyst to access all the global FORTRAN variables in HOS and thus they possess all the power inherent in a high-level computer language like FORTRAN. As a result, they can be used for other purposes than to simulate the psychologically plausible operations of arithmatic and logic. The open-endedness of operator functions poses some real difficulties when we attempt to interpret just what cognitive operations are represented by any specific operator function. Such uses of operator functions need not, however, compromise the accuracy of HOS, if the user is careful to assign a zero time cost to any operator function that does not represent a cognitive process or to assign a high enough time cost to any function that represents more than a simple mental calculation.

Because of the diversity of mental processes that can be simulated by operator functions, the analyst must define each operator function in FORTRAN and supply a basic time cost for each function. Whenever the value of the function is required by a procedure or by another operator function, HOS will either attempt to remember the value of the function, or will execute the appropriate code. The time associated with the function calculation will be the sum of the times required to obtain any values needed by the function (e.g., values of devices or other functions), plus the basic time cost for the function multiplied by the number of times the hab strength calculations must be repeated before any one of the following criteria is satisfied.

- Hab strength is greater than or equal to a threshold "confidence" level (Equation 6 in Section A.6.2).
- Two successive determinations of the hab strength differ by less than a user-specified amount (Equation 7 in Section A.6.2).
- The number of iterations multiplied by the basic time cost exceeds a user-specified time limit.
- The number of iterations exceeds the user-specified limit.

In determining the time costs for operator functions, the user will have to depend mainly on his own ingenuity and resources because of the sparsity of general guidelines and experimental data in the published literature. Whenever possible, it is advisable to conduct at least an informal experiment to determine an approximate time cost. For general theories on mental arithmetic and some relevant experimental data, we refer the reader to Thomas (1963), Dansereau and Gregg (1966), and Restle (1970). Some time-and-motion study data on mental work is available in Quick, Duncan, and Malcolm (1962).

A.6.4 The Absorption Model

The HOS model of perception is tailored specifically to the information processing requirements of a complex man-machine interface. Whereas most contemporary theories and models for perception attempt to describe how sensory signals are processed into higher order codes, the HOS absorption model is concerned only with the changes through time in the operator's knowledge of the state of a display or control. It is therefore best to refer to it as an absorption model rather than a perception model.

The same model is used to describe the absorption of information both by vision and by touch. The user must specify which modality is appropriate for absorbing from each device and whether the other modality can be used when the preferred channel is otherwise occupied. The user must also provide HOS with a single parameter value for each device that specifies a basic time cost unit for absorptions from the device.

Absorption occurs in quantum steps called micro-absorptions that are repeated until a termination criterion is satisfied. Each micro-absorption consists of an updating of the estimated value and the hab strength of an item and the assessment of a time charge for the micro-absorption. The micro-absorption process is repeated until any of the following four termination criteria is satisfied (basically the same as for function computations):

(1) Hab strength is greater than or equal to a threshold "confidence" level (Equation 6 in Section A.6.2).

(2) Two successive determinations of the hab strength differ by less than a user-specified amount (Equation 7 in Section A.6.2).

(3) The amount of time spent in the absorption process exceeds a user-specified time limit.

(4) The number of iterations exceeds the user-specified limit.

This process is illustrated in the flowchart in Figure A-6 in which T denotes the simulation time, C is the micro-absorption time charge, and H(k), V, R, K, d, and L are the quantities described in Section A.6.2.

The details of these updating and time assessment processes are determined by whether the device is discrete, continuous, or positional; accordingly, we will analyze each case separately.

For discrete devices, the estimated value of a device is set equal to its actual value, i.e., absorption of discrete devices is error-free. The hab strength for discrete devices is modified according to the general formula described in Section A.6.2, with the similarity between the last two estimates being set to 1.0. The time charge for each micro-absorption is simply the basic time charge associated with the device.

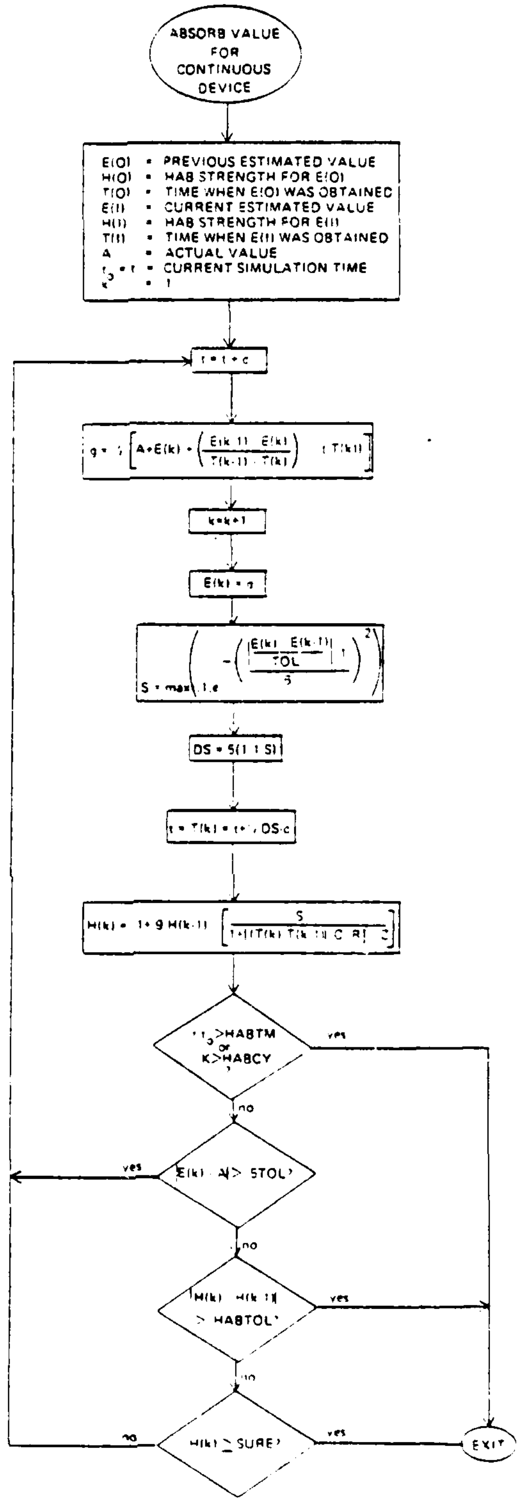

ABSORB VALUE FOR CONTINUOUS DEVICE
E(O) = PREVIOUS ESTIMATED VALUE
H(O) = HAB STRENGTH FOR E(O)
T(O) = TIME WHEN E(O) WAS OBTAINED
E(1) = CURRENT ESTIMATED VALUE
H(1) = HAB STRENGTH FOR E(1)
T(1) = TIME WHEN E(1) WAS OBTAINED
A = ACTUAL VALUE
t₀ = t = CURRENT SIMULATION TIME
k = 1
t = t + c
g = ½ [A + E(k) + ((E(k-1) - E(k)) / (T(k-1) - T(k))) (t - T(k))]
k = k + 1
E(k) = g
S = max(.1, e^(-((|E(k) - E(k-1)| / TOL - 1) / 5)²))
DS = 5(1 - S)
t = T(k) = t + ½ DS·c
H(k) = .1 + .9 H(k-1) [S / (1 + |(T(k) - T(k-1)) C - R| - C)]
t - t₀ > HABTM or K > HABCY ?
yes
no
|E(k) - A| > 5TOL ?
yes
no
|H(k) - H(k-1)| > HABTOL ?
yes
no
H(k) ≥ SURE ?
no
yes
EXIT

Figure A-6. HOS Absorption Model

Since the model automatically sets the estimated value equal to the actual value on the first micro-absorption, the model makes no attempt to describe the relative accuracy of the absorptions from discrete devices. Thus, absorption from a discrete device is completely deterministic.

The absorption process for continuous and positional devices is similar to that for discrete devices, with embellishments appropriate to the fact that the devices are continuous. The major differences are that:

(1) On each micro-absorption a new estimated value is determined by averaging the actual value with a value extrapolated linearly from the last two estimates.

(2) The micro-absorption process is not allowed to terminate until the estimated value lies within a user-prescribed range around the actual value (the "accuracy" associated with the device).

(3) The time cost for each micro-absorption is calculated from the basic micro-absorption time charge for the device and the "dissimilarity" between the last two estimated values.

Absorptions from positional devices (i.e., absorptions of ordered pairs of numbers) are treated as simultaneous absorptions of the two numbers in the ordered pair, each number being treated as a continuous value. The time charge for an absorption is then the maximum of the time costs for the individual components.

It is difficult to derive a general formula for the time cost of an absorption from a continuous or positional device, because of difficulties in specifying the time cost for each iteration or the number of iterations that will be required to bring the estimated value within the specified tolerance range. As indicated in Figure A-6, the time cost of each iteration cycle is $(1 + \frac{1}{2} \cdot \mathrm{DS})C$, where

$$\mathrm{DS} = \min \left[0.5,\ 1 - e^{-\left(\frac{\left| \frac{E-PE}{TOL} \right| - 1}{6} \right)^2} \right]$$

is a measure of the dissimilarity between the two most recent estimates of the device and E and PE are the two most recent estimates. Note, that if $|E-PE| \leq TOL$, then DS = .5. That is, if the two most recent estimates differ by less than the "accuracy," then the dissimilarity value is exactly .5. It is likely that this condition will hold for most absorption processes, at least after the first iteration cycle, so the time cost for each such micro-absorption will be exactly 1.25C. Given this situation, the recursion formula for E(k+1), reduces to:

$$E(k+1) = .5A + .9E(k) - .4E(k-1)$$

Two sample sequences of estimated values are graphed in Figure A-7; it is assumed that the actual value, A, is constant and that $|E(1) - E(0)| \leq TOL$. Observe how the values are first on one side of the actual value and then are on the other side. Also, note that the speed of convergence of the sequence does not differ very much between the two sequences even though the two initial values are quite different in the two cases. We have studied many such sequences of estimated values generated by the absorption model in order to determine just how much the rate of convergence actually varies. Figure A-8 displays the number of iterations that must be performed for a variety of initial values in order to bring the estimated value within a two percent tolerance range of the actual value. Although other tolerance values will generally be used in HOS, it is interesting to note that convergence to this fairly strict tolerance is rapid and varies only slightly between even the most disparate choices of initial values.

A.6.5 The Short-Term Memory Model

he HOS operator is considered to be a "trained" operator. Thus, there are many quantities (e.g., knowledge of task procedures and locations of stationary displays and controls) that are considered to be in long-term memory, i.e., the operator's ability to recall any of these quantities is unaffected by the passage of time. However, most of the display and control

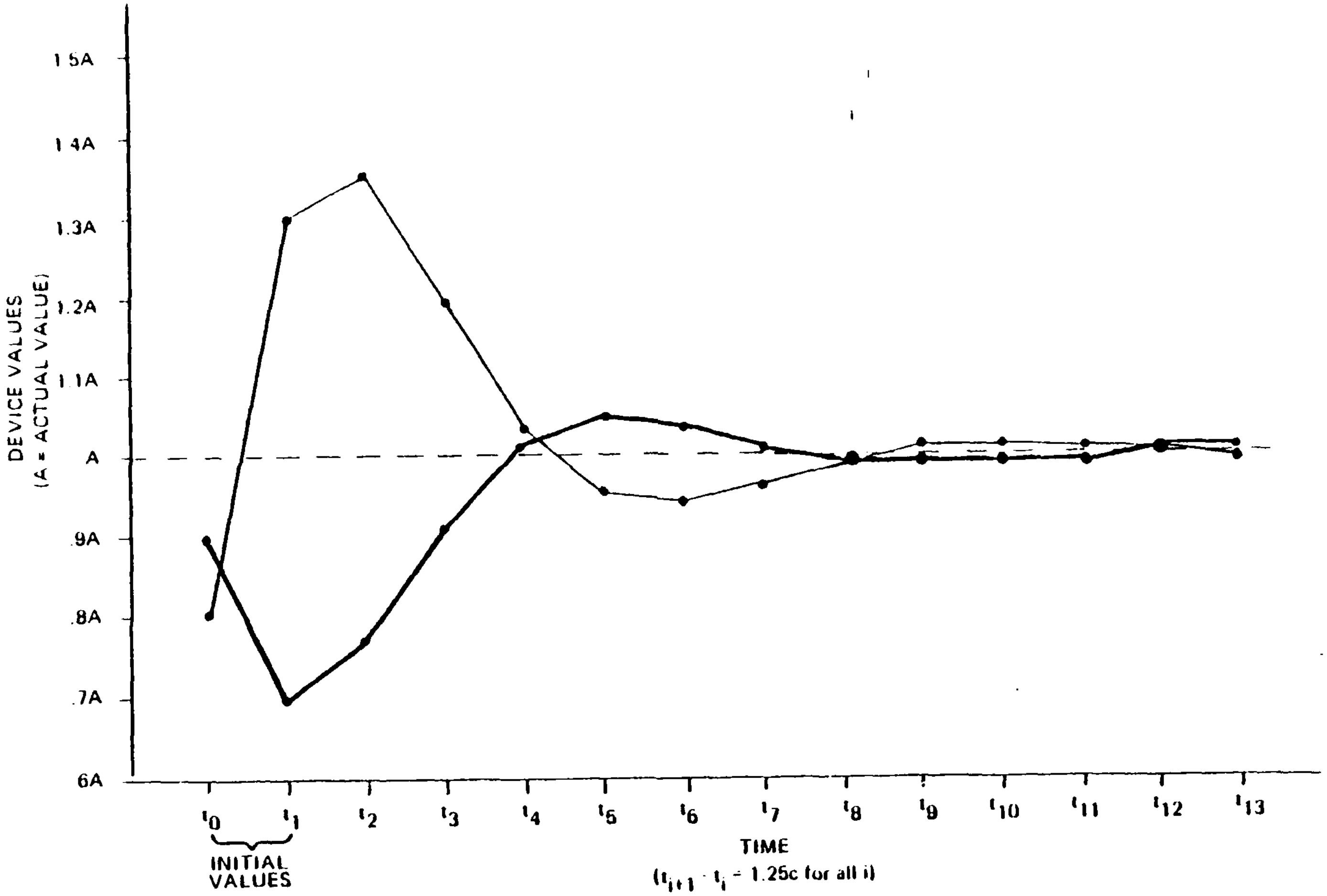


Figure A-7. Convergence of Estimated Values

$\frac{E(1)}{A}$	.6	.7	.8	.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6
1.6	7	7	7	7	3^7	3^7	7	7	6	6	6
1.5	7	7	7	7	3^7	3^7	7	6	6	6	2^6
1.4	7	7	7	3^7	3^7	3^7	6	6	2^6	2^6	6
1.3	7	7	7	3^7	3^7	3^6	6	2^6	5	5	5
1.2	7	7	7	3^6	3^6	3^6	2^5	5	1^5	5	5^7
1.1	4^7	4^7	4^6	3^6	3	2	1^4	4	4	4^7	4^7
1.0	4^7	4	4	3	1	3	4	4	4^7	4^7	4^7
.9	4	4	1^4	2	3	3^6	4^6	4^7	4^7	4^7	4^7
.8	5	5	2^5	3^6	3^6	3^6	7	7	7	7	7
.7	5	2^6	6	3^6	3^7	3^7	7	7	7	7	7
.6	6	6	6	3^7	3^7	7	7	7	7	7	7

$\frac{E(O)}{A}$

$$E(k+1) = \tfrac{1}{2}\left[A+E(k) + \left(\frac{E(k)-E(k-1)}{T(k)-T(k-1)}\right)\cdot(t-T(k))\right]$$

$$= \tfrac{1}{2}\left[A+E(k) + \left(\frac{E(k)-E(k-1)}{1.25c}\right)\cdot c\right]$$

$$= .5A + .9E(k) - .4E(k-1)$$

Figure A-8. Number of Extrapolation Iterations Required to Bring Estimated Value Within a Two Percent Tolerance of Actual Value (A)*

*If any subsequent iteration falls outside two percent bounds, superscript indicates first iteration such that no subsequent iterations exceed tolerance.

values vary with time. These items are normally not committed to long-term memory and, therefore, the HOS process of retrieving the data corresponds to a short-term memory model.

The variety of human performance characteristics that a short-term memory model should be able to describe and that have been addressed by other models of memory, include:

- Probability of successful recall.
- Time cost (or latency) or recall.
- Probability of transfer to long-term memory.
- Effects of interactions between items in short-term store and items in long-term store.

The HOS memory model has been designed to predict only the probability of recall and the time cost associated with a recall attempt. HOS considers only the estimated value of a device to be subject to decay and forgetting -- none of the other characteristics of a device enter the recall model -- i.e., no other characteristics of a device can be forgotten. HOS will, at the option of the user, extrapolate recalled values to the time at which the recall attempt is made and/or degrade the accuracy of recalled values to simulate the uncertainty associated with the recalled value.

The HOS memory retrieval model is based on data obtained in several experimental studies of short-term memory,* in which it was found that the relationship between probability of correct recall of an item and time since presentation, could be described by:

$$P = H^{\sqrt{t}} \qquad (8)$$

*Peterson and Peterson (1959) and Murdock (1961).

where

P = the probability of correct recall.

t = the time interval between presentation and attempted recall.

H = a constant in the unit interval characteristic of the subject and the experimental situation.

The H in Equation 8 is the formal definition of hab strength. However, as described in the preceding sections, rather than being a constant, H is modified by HOS whenever a device or function is estimated.

Equation 8 was found to hold for a variety of experimental situations, providing that the subjects are prevented from rehearsing the stimuli during the period between presentation and recall. Notice that this equation implies that if $H \neq 1$, P must approach zero as t becomes large, so that the processes of long-term memory and random guessing are not described.

It should be noted that the function usually used in psychological literature to summarize the short-term memory data, differs slightly from the previous equation. The function is generally described as exponential (Pollatsek, 1969) with the general form:

$$P = Ke^{-kt} \qquad (9)$$

where K and k are positive constants. Equation 8 differs from this general form only in that the exponent is $\sqrt{t}$ rather than t. This particular choice of exponent was made in order to make the memory model produce reasonable recall probabilities when the hab strengths were generated by the procedure described in Section A.6.2. A comparison of Equations 8 and 9 with the experimental data, has revealed that neither function produces a significantly better fit than the other.

The HOS short-term memory model is actually somewhat more complicated than Equation 8 would indicate. This equation is used to obtain the probability of successful recall on a single recall attempt. However, under certain conditions, the operator can make more than one recall attempt and the time consumed by the recall process depends both on the number of attempts and on whether or not recall succeeds. In addition, the model has optional mechanisms for degrading the precision and extrapolating the values of continuous devices that are successfully recalled.

Figure A-9 describes the complete HOS memory model. The time consumed by an attempt to recall a value is indicated as "Cost" in the figure. Each micro-attempt at recall (i.e., each cycle through the model) requires a constant amount of time and if recall succeeds, the time cost is incremented one additional time by the same amount ot account for the time spent in retrieving the preceding value (i.e., the next to the last value which is needed to extrapolate to the current estimate). The short-term memory cycle time, R, is an input parameter that is constant for each simulation and considered to be a characteristic of the operator.

If a micro-attempt at recall fails (i.e., if X, a random number drawn from a uniform distribution on the unit interval, is greater than P, the probability of recall), then further micro-attempts may or may not be made, depending on the value of (X - P)/H. If this value is less than or equal to a user-specified constant, then another micro-attempt will be made.* This formalizes the intuitive notion that the operator will continue to attempt to recall a value if the current recall attempt has almost succeeded.

*If limits on the number of recall attempts and the amount of time spent in recall have not been exceeded.

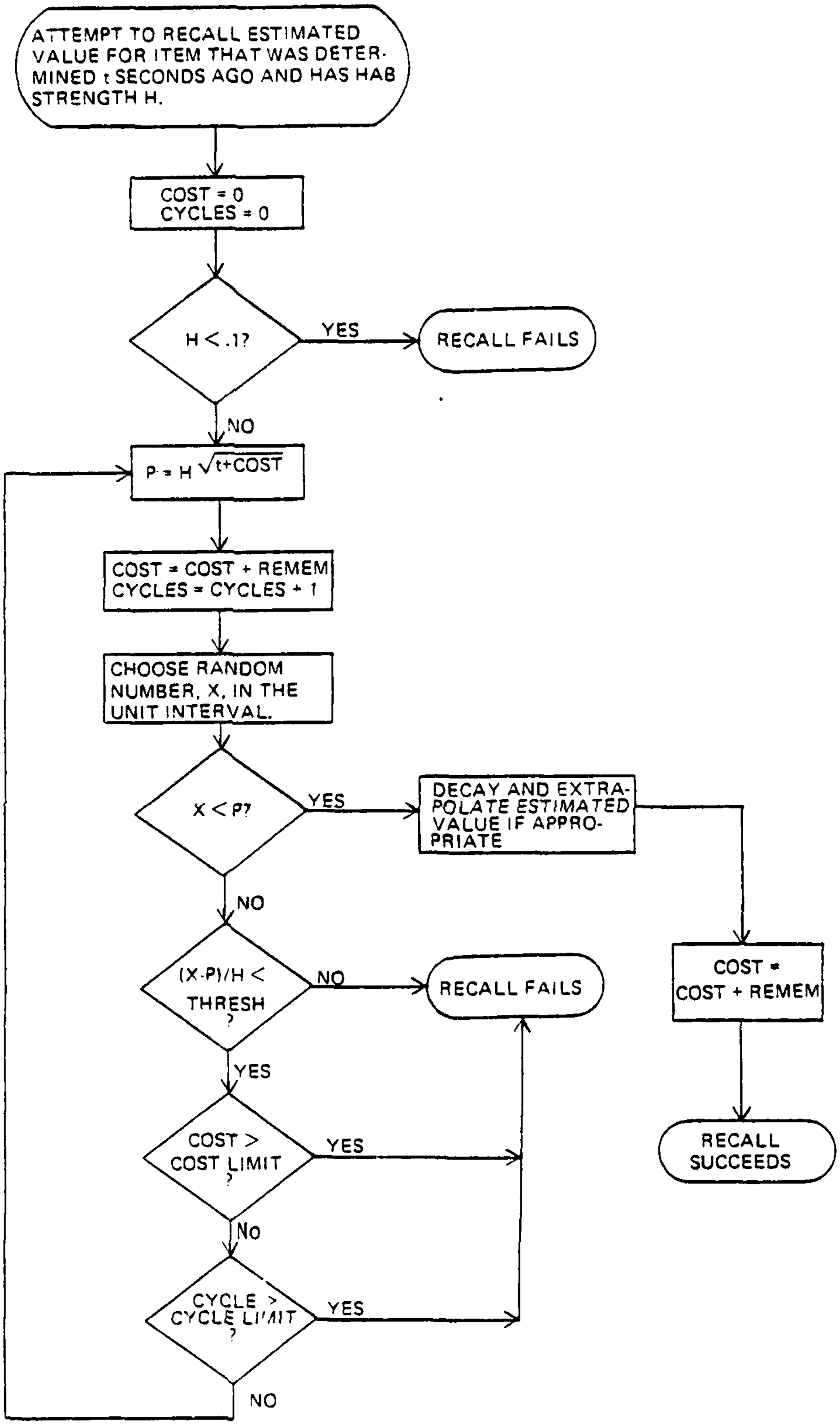


Figure A-9. HOS Short-Term Memory Model

Since the probability of repeated micro-attempts at recall is dependent on the user-supplied values, the possibility of getting "stuck" in a perpetual recall loop exists. For example, if the hab strength, H, for an item is .5, and if 100 seconds have elapsed since a value was last estimated (i.e., t = 100), then the probability of successful recall at each micro-attempt is at most

$$P = H^{\sqrt{t}} = .5^{\sqrt{100}} = .001$$

If the value of d were 2.0, then whenever a micro-attempt failed (as would happen more than 99.9 percent of the time), another micro-attempt would always be allowed since

$$\frac{X-P}{H} = \frac{X-P}{(.5)} \leq 2.0$$

which must be true because $0 < X\text{-}P < 1.0$. Thus, although there would be virtually no chance of recalling the value, the operator would never stop trying. HOS provides automatic checks on the recall probabilities and the user can supply limits on the total amount of time and number of recall attempts to be allowed to prevent such loops. If any of these limits are exceeded, recall is assumed to have failed.

Although the basic recall model described by Equation 8 is analytically manageable for purposes of parameter estimation, the complete recall model is not. Although it has not been possible to derive explicit expressions for the probability of successful recall and amount of time consumed in the recall process under all conditions, reasonable approximations have bee, obtained for the two cases of most interest. These two situations cc-respond to cases in which:

(1) The recall process is dominated by the input variable d.

(2) The recall process is dominated by the cycle and cost limits.

For both cases, it is assumed that $0.1 \leq H < 1.0$, since the predictions of the model are obvious for other values of these variables. In this discussion, the following notations will be used:

H = hab strength for the item of interest.

t = time since the estimated value for the item of interest was last obtained.

$N = \min\left(\text{cycle limit}, \frac{\text{cost limit}}{R}\right) =$ maximum number of micro-attempts allowed.

P_i = probability that recall succeeds on the i^{th} micro-attempt.

q_i = probability that recall fails and further micro-attempts are prohibited after the i^{th} micro-attempt.

r_i = probability that recall fails and further micro-attempts are allowed after the i^{th} micro-attempt.

P = probability of eventual successful recall.

Q = probability of eventual failure to recall.

R = short-term memory cycle time.

d = user-supplied tolerance such that further recall attempts will be permitted if the random selected variable, $X > H \sqrt{t}$.

C = total time required to obtain the estimated value.

A = time cost of obtaining the estimated value of the item of interest by absorption or computation.

S_i = time cost of the recall process given that it succeeds after exactly i micro-attempts.

F_i = time cost of the recall process given that it fails and exits after exactly i micro-attempts.

Note that some of these variables represent constants and others should be considered as random variables. In particular, C, A, and F_i will be treated as random variables. Our main interest will be in their respective expected values, denoted as $\overline{C} = E(C)$, $\overline{A} = E(A)$, and $\overline{F_i} = E(F_i)$.

Case 1: D Dominated Model

In this case, d is such that the recall process is terminated by the random variable X exceeding $H^{\sqrt{t}} + d \cdot H$. This condition can be stated explicitly as:

$$\Pr\left(\frac{X - H^{\sqrt{t + (i-1)R}}}{H} > d\right) > 0 \text{ for } i=1, 2, \ldots, N \qquad (10)$$

where X is a random variable that is uniformaly distributed on the unit interval and $H^{\sqrt{t + (i-1)R}}$ is the probability of recall succeeding on the i^{th} micro-attempt, given that the process does not terminate on an earlier micro-attempt. For $.1 < H < 1.0$ and $R > 0$, it must be true that $H^{\sqrt{t}} > H^{\sqrt{t + (i-1)R}}$ for $i > 1$. Therefore, for any X,

$$\frac{X - H^{\sqrt{t + (i-1)R}}}{H} \geq \frac{X - H^{\sqrt{t}}}{H} \quad \text{for } i=1, 2, \ldots, N$$

and Equation 10 may be rewritten as:

$$\Pr\left(\frac{X - H^{\sqrt{t}}}{H} > d\right) > 0 \qquad (11)$$

Equation 11 is satisfied if and only if the condition within the parenthesis is valid for X = 1. Hence, our restriction on the magnitude of d reduces to:

$$0 \leq d < \frac{1 - H^{\sqrt{t}}}{H} \qquad (12)$$

Clearly, we cannot choose d so that Equation 12 will hold for all admissable values of H and t except by choosing d = 0, which is not interesting, so we must consider what sort of constraints Equation 12 places on H and t. Figure A-10 indicates the solution to this problem for six different values of d. So long as the point determined by H and t is below the line in the figure for a given value of d, then Equation 12 will hold for those values of H, t, and d. Since hab strengths in HOS will typically be in the vicinity of .9 and recall intervals will frequently be as short as one to two seconds, d must be approximately .10 in order for Equation 12 to be valid and for Case 1 to be applicable.

Under the assumptions that Equation 12 holds, that R is small in comparison to t and that N is large, an approximation for P, the probability that the recall process will ultimately succeed can be obtained. The probability of success, P_i, on the i^{th} micro-attempt at recall is given by the formula:

$$P_i = d^{i-1} \cdot H^{i-1} + \sqrt{t + (i-1) R}$$

Thus, the probability of eventual successful recall, P, is the sum of the success probabilities for all possible micro-attempts.

$$P = \sum_{i=1}^{N} P_i = \sum_{i=1}^{N} \left[d^{i-1} \cdot H^{i-1} + \sqrt{t + (i-1) R} \right]$$

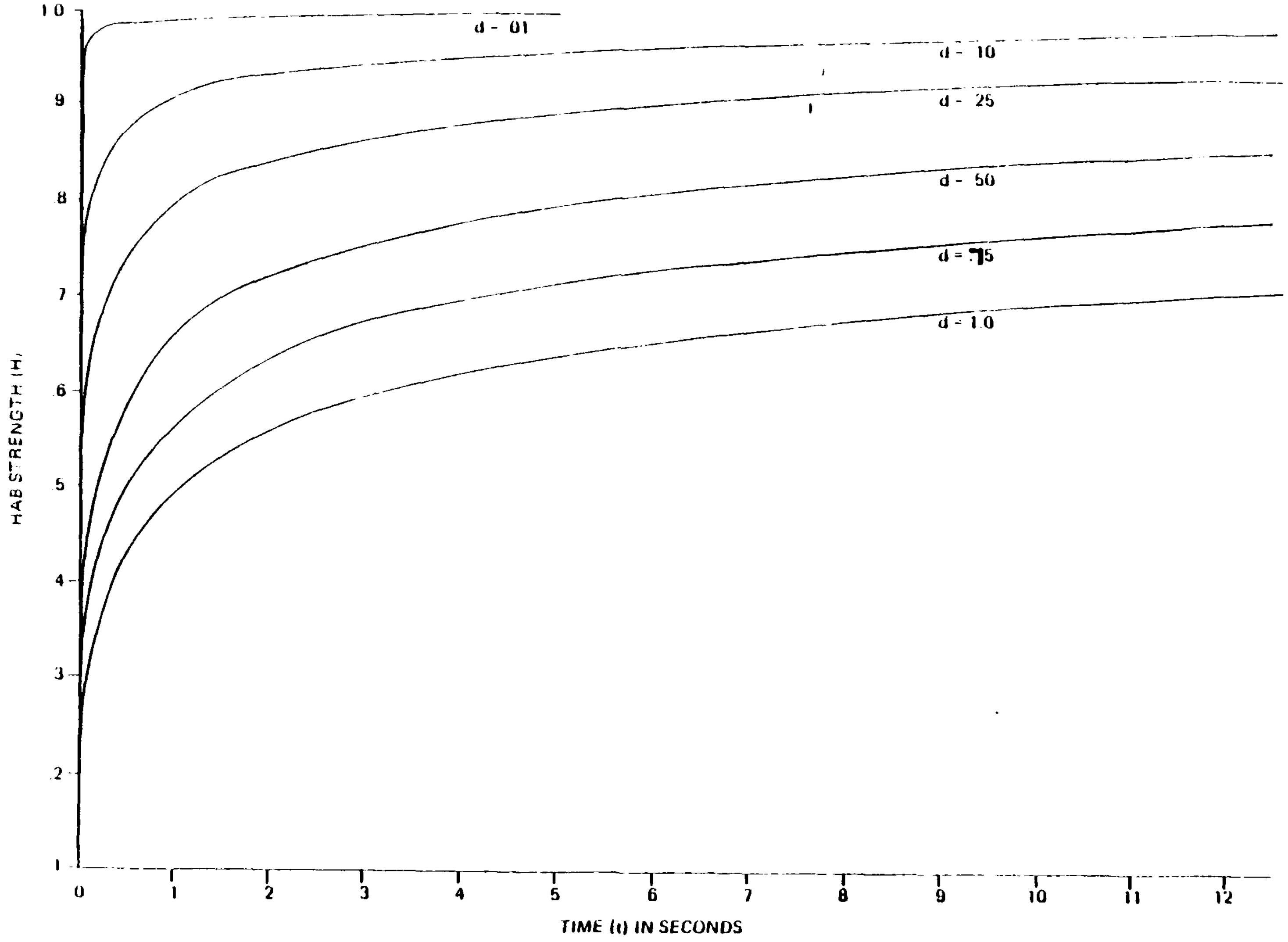


Figure A-10. H as a Function of d and t for Case 1

Invoking the assumption that R is small in comparison to t yields:

$$\hat{P} = H\sqrt{t}\left(\frac{1 - d^N \cdot H^N}{1 - d \cdot H}\right) \qquad (13)$$

where $\hat{P}$ is an estimate of P.

As $N \rightarrow \infty$ (i.e., if the cost and cycle limits are removed from the recall process) Equation 13 reduces to:

$$\hat{P} = \frac{H\sqrt{t}}{1 - d \cdot H} \qquad (14)$$

The estimate, $\hat{P}$, of P in Equation 14 is actually without error if $N = \infty$ and $R = 0$. When these assumptions are not met, however, it is desirable to know how close we can expect $\hat{P}$ to be to P. Under the conditions that $N = \infty$ and $R \geq 0$, it can then be shown that the accuracy of Equation 14, as an estimate for P, is constrained by the following inequality:

$$0 < \hat{P} - P < \frac{\ln\left(\frac{1}{H}\right) \cdot R \cdot d \cdot H}{2\sqrt{t}\left(1 - d \cdot H\right)} \hat{P}$$

Notice, in particular, that $\hat{P}$ will always be an over-estimate of P and the estimation error will approach zero as the ratio $\frac{R}{\sqrt{t}}$ approaches zero.

Turning now to the estimation of the time-cost of the recall process for Case 1, we observe that the mean total time cost, $\overline{C}$, for the process of obtaining an estimated value for an item can be written as:

$$\overline{C} = P\overline{C}_s + (1 - P)\ \overline{C}_f \cdot$$

where the mean time-cost for recall processes that succeed is:

$$\overline{C}_s = \sum_{i=1}^{N} P_i\ \overline{S}_i$$

and the mean time-cost for recall processes that fail is:

$$\overline{C}_f = \sum_{i=1}^{N} Q_i\ \overline{F}_i$$

and where

$$\overline{S}_i = \left(i + 1\right) \cdot R$$

is the time-cost of the process if recall succeeds on the i^{th} micro-attempt and

$$\overline{F}_i = E\left(F_i\right) = i \cdot R + \overline{A}$$

is the mean time-cost if recall fails and the process terminates on the i^{th} micro-attempt.

AD-A094 353   ANALYTICS INC WILLOW GROVE PA   F/G 5/8
THE HUMAN OPERATOR SIMULATOR. VOLUME IX. HOS STUDY GUIDE.(U)
SEP 78 M I STRIEB, F A GLENN, R J WHERRY   N62269-78-M-6684
UNCLASSIFIED   TR-1320-VOL-9   NL

4 OF 4
AD-A
094353

END
DATE
FILMED
2 81
DTIC

Again, assuming that Equation 12 holds, that $N = \infty$ and that t is large with respect to R, we can obtain the following approximations for $\bar{C}_s$ and $\bar{C}_f$:

$$\hat{C}_s = R \cdot \left(\frac{\hat{P}}{H\sqrt{t}} + 1 \right) . \qquad (15)$$

and

$$\hat{C}_f = \frac{R \cdot \hat{P}}{H\sqrt{t}} + \bar{A}$$

where $\hat{P}$ is the approximation to P given by Equation 14.

It can also be shown that, under the above stated conditions, *the variance, V_s, of the cost for successful* recalls is approximated by the formula:

$$\hat{V}_s = \frac{R^2 \cdot d \cdot H}{(1 - d \cdot H)^2}$$

Case 2: Recall Model Dominated by Cost and Cycle Limits

At the opposite extreme from Case 1, are situations in which d is sufficiently large that an additional micro-attempt at recall is always allowed after the failure of one micro-attempt. This condition can be stated more formally as:

$$Pr\left(\frac{X - H\sqrt{t + (i-1)R}}{H} > d \right) = 0 \text{ for } i=1, 2, \ldots, N \qquad (16)$$

Assuming that Equation 18 holds and that t is large in comparison to N · R, then

$$\hat{P} = 1 - \left(1 - H\sqrt{t}\right)^{N} \tag{19}$$

This is just the probability that recall will not fail on any of N independent micro-attempts when the probability of failure for each micro-attempt is assumed to be $1 - H^{\sqrt{t}}$.

Similarly, it can be shown that:

$$\hat{C}_s = \frac{R\left[1 + H\sqrt{t} + \left(1 - H\sqrt{t}\right)^{N} \cdot \left(1 - (N + 1)\ H\sqrt{t}\right)\right]}{H\sqrt{t}\left[1 - \left(1 - H\sqrt{t}\right)^{N}\right]} \tag{20}$$

and

$$\hat{C}_f = N \cdot R + \bar{A} \tag{21}$$

are adequate approximations for the time-costs for successful and unsuccessful recall processes.

Experimental Comparisons

For the purpose of comparing model predictions with experimental data, it is necessary to identify the procedural features of a memory experiment in which either the Case 1 or Case 2 derivations would apply. Case 1 would seem to apply to experiments for which the subject is encouraged to admit failure whenever an initial recall attempt is not at least almost successful (the condition described in Equations 10 through 12) and for which a substantial time is allowed for recall (the condition that $N \rightarrow \infty$). For Case 2, a relevant experiment would be one in which the subject is encouraged to make repeated attempts at recall until he succeeds or until a time limit that is small in comparison to the retention interval elapsed. Also, to validate the models, experiments that employ numerical information as the object of recall are most relevant. In addition, the experiment must include some method for preventing the subject from rehearsing the test items during the retention interval.

Unfortunately, we have failed to find any short-term memory studies that satisfy these constraints. The primary problem is finding experiments dealing with recall of numerical information. The only study which we located that included such experiments was one performed by Cohen (1971). Unfortunately, Cohen did not report response latencies and her precedures cannot be characterized by either of the two cases for which we have approximate performance predictions for the HOS model. (Her retention intervals were between 5 and 20 seconds, while her subjects were allowed 10 seconds to attempt to recall each item and they were encouraged not to stop trying until the time limit elapsed.) We were able to determine, however, that the frequencies of correct recall obtained in Cohen's experiments were generally consistent with the basic recall function ($P = H^{\sqrt{t}}$), upon which the HOS memory model is based.

It is interesting to note that the response latencies for successful recall processes for the HOS model depend on the retention interval in both cases for which time-cost approximations were derived. Since that dependence assumes a rather complicated mathematical form (Equations 15 and 20), we have determined some illustrative empirical relationships between retention interval and time-cost for successful recall processes. Figure A-11 portrays these results. The simulation that produced these data modeled the recall of consonant trigrams, with each trigram being treated as three separate items. It seems somewhat surprising that response latency in the figure is virtually constant over a large range of retention intervals. This observation is consistent with the claim of Waugh (1969) that, for verbal material, the mean latency of successful recall from short-term memory, is independent of retention interval. Some further simulation results for the same consonant trigram memory model are presented in Figure A-12, together with some experimental data. Note that an optimal fit to the results of Peterson and Peterson (1959) and Murdock (1961) is achieved when the parameter REMEM is set to .07. Since these data inspired the basic function for recall upon which the HOS memory model is based, it is reassuring to note that the elaborated model continues to fit the same data. We imaging that the disparate results of Melton, Crowder, and Wulff (1963), also displayed in Figure A-12, are a consequence of factors in their experiment which allowed their subjects to commit the test items to long-term memory. Accordingly, we are not particularly concerned with the failure of any of our HOS simulations to mimic those results.

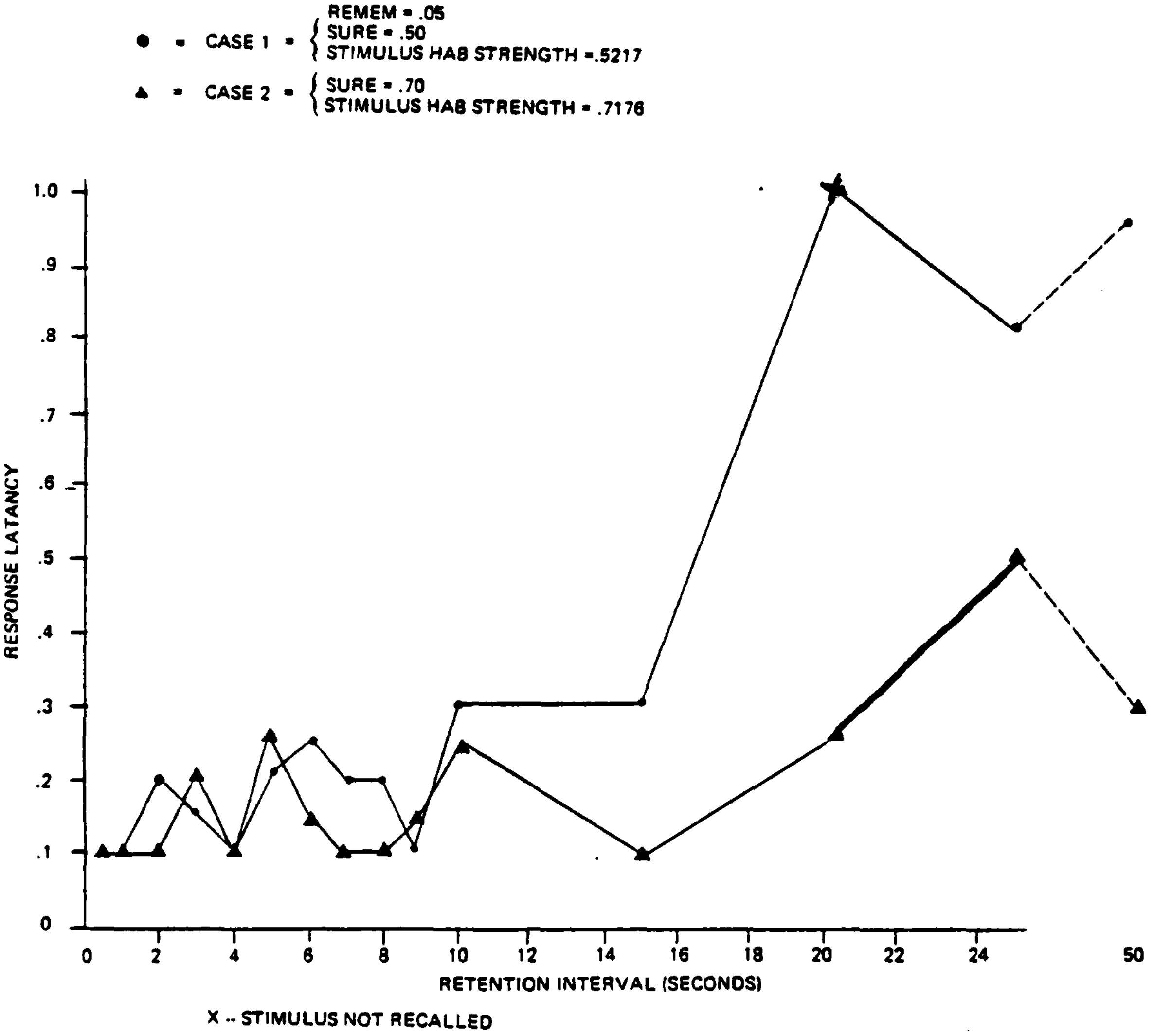


Figure A-11. Simulated Response Latencies for Successful Recall as a Function of Retention Interval (Consonant Trigrams)

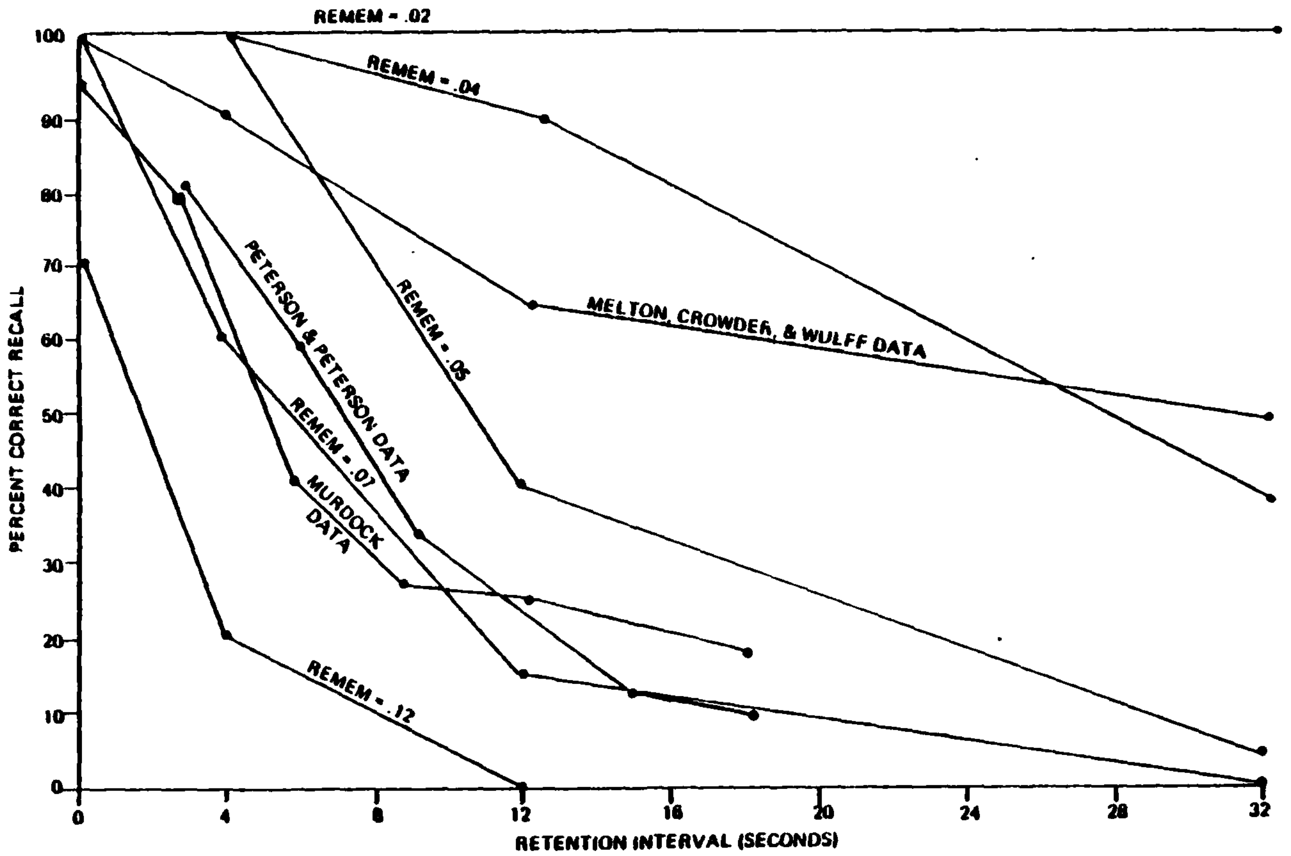


Figure A-12. Comparison of Experimental and Simulated Results of Recall of Consonant Trigrams

APPENDIX B
A BRIEF HISTORICAL PERSPECTIVE OF HOS
(1967-1978)

APPENDIX B
A BRIEF HISTORICAL PERSPECTIVE OF HOS
(1967-1978)

Robert J. Wherry, Jr., PhD.

It is a truism that necessity is often the mother of invention -- and this is certainly true with regard to HOS. It was conceived out of feelings of frustration and disappointments with the impotency of human engineering technology of the mid-Sixties. The concept of a Human Operator Simulator (HOS) did not suddenly appear to me one day, but was, I believe, the inevitable outcome of consciously searching for a better approach to solving human engineering problems. The concept of modeling human behavior had attracted me for a number of years, however, prior to those months in 1967 when HOS was ultimately conceived. I am certain that the prior work with which I had been involved in the area of vigilance behavior, information processing under stressful conditions, and predictions of student pilot success or failure were instrumental in directing the ultimate conception of how humans processed information and did various tasks. Factor analytic studies I had done in Pensacola, Florida on a rather wide variety of pilot tasks had left on me an indelible appreciation (or belief, at any rate) that, perhaps, only a few independent factors really accounted for goodness of performance in what, at first, had appeared to be very diverse tasks. Finally, the experience which I had gained since 1959 in programming computers for complex applications in aviation psychology, medicine, and biophysics had made a believer of me with regard to the potential power of computer simulation for solving all sorts of problems.

Thus, HOS not only developed from a specific need, but it also grew out of what I consider to be an unusual and fortuitous series of experiences to which I had been exposed. To better appreciate the specific purposes for which HOS was initially conceived and developed, I must take you back to late 1966 when I was transferred from Pensacola, Florida to the Naval Missile Center (NMC) at Point Mugu, California to head up the human engineering branch. Our mission there was to accomplish the tests and evaluations of new Naval airborne weapons systems.

To perform a test and evaluation one must, of course, first decide what one desires to test. It became obvious that two different approaches were possible. The first I shall refer to as "comparison with specs and standards" and the second I shall call "performance evaluation." The first approach dealt with testing whether various aircraft displays, controls, labels, panels, etc., conformed to Human Engineering guides, standards and specifications. It may be recalled that MIL STD 1472 and MIL SPEC 46855 were first issued in 1966. Because of this we had in our possession, at that time, the latest documents containing data on what the Navy (and the other services as well) deemed to be "acceptable" HE design standards. On the other hand, because of the newness of those documents, no system arriving for test and evaluation at NMC for several years thereafter would have required a contractor to meet those standards and specifications. Thus, those documents did offer a standard of comparison by which at least some aspects of the crewstations could be evaluated even though it might be difficult or impossible to force an air frame contractor comply with those standards. A second drawback in using MIL STD 1472 was that no guidance on the impact on operator or system performance was provided in cases where various aspects of crewstation design failed to meet the new standards. I found that it was virtually impossible to get the Navy interested in correcting any single deficiency, because no single deficiency was ever so bad as to be able to say that it alone made the aircraft either unsafe or that it alone would be the cause of unsuccessful or aborted mission performance. It was obvious to our human engineering team that *the cumulative effect of a series of minor deficiencies could and would have a major impact on system safety and mission success.* To be able to convince others of this point of view, however, would require a fairly detailed model of the impact of various display and control features on human information absorption, processing, and transmission in a task sequencing framework to illustrate such cumulative effects! Unfortunately, such models were not available at that time.

The second approach to the test and evaluation of the crewstation dealt with attempting to determine (regardless of conformance or non-conformance to various MIL STDs) if the operators were able to adequately perform the various functions which had been allocated to them. In attempting to determine precisely what was expected of a given operator, we had occasion to examine a wide variety of task analyses and timelines which had been prepared by a variety of different contractors. Without exception, these rather costly items, when they had, in fact, been prepared, were extremely disappointing in terms of adequately expressing what was actually expected of a given operator. All too often task analysis blocks had been prepared at a very macro level (e.g., "Pilot acquires and locks on target") and times assigned to such activities were,obviously, merely "educated guesses." It was my personal experience that, at least by the time a weapon system was delivered to NMC, no task analysis or timeline indicated that the operator would be too busy to perform all the functions he had been assigned. The task analyses which we reviewed in those days also failed to give the reader a good appreciation of the often necessary simultaneity of various different task demands facing a particular operator during crucial segments of a mission. It became obvious that a more stringent set of rules were needed in guiding whoever prepared task analyses so that (a) an appropriate level of detail would be included, and (b) a given statement made by a task analyst could be interpreted without ambiguity as to what the operator's responsibilities were. (From this concept, the Human Operator Procedures (HOPROC) language ultimately arose.) Further, it was felt that a successful accomplishment of any task analysis really involved two distinct efforts, the first of which was expressing what was expected of the operator (in terms of what actions he must take) and the second was (given the displays, controls, and layout of the crewstation) to determine if the operator could, in fact, accomplish all those assigned tasks within the requisite time. This implied that the tasks themselves ought to be able to be described independently of the particular crewstation layout, *and, if a sophisticated human performance model were available,* then the impact of different crewstation designs could be reliably and objectively evaluated without relying on "educated guesses" by contractor personnel who had a personal interest in making their own aircraft appear to be good in the Navy's eyes.

In a very real sense, the first concept of HOS was never intended to simulate all types of human performance, but it did set out to quantify performance times of various types of anatomy movement (head and eyes, hands and arms, feet, etc.) and the effect of various features of displays and controls (e.g., size, contrast, shape, etc.). Actually, my own feeling by late 1967, was that we were a long way from being able to predict the times various mediated mental processes might take, but that at least those observable events, such as anatomy movements, absorption of information from displays, and manipulation of controls, should be able to be accurately predicted. In this respect, I was especially encouraged by the work of Topmiller and Sharp (1965) which had indicated that arm-hand reach time was very predictable. Also several informal studies (which, I deeply regret, have never been published) on eye movement and fixation times and on numeral and dial reading times which were conducted by Alvah Bittner and myself at Pt. Mugu that greatly supported the concept that any task could be broken down into sequences of various "micro" processes and the sum of the micro process times would, in fact, yield the total task times. Many people rejected such a hypothesis and predicted that there would be tremendous interactions among many if not all of the micro processes which would make the analytical "additive" approach I was advocating doomed to failure. Such discussions and arguments, I might add, were very philosophical, since neither I nor my opponents had sufficient data to support our contentions in those days.

I suppose I stuck with the belief that each micro process was independent primarily because, if it turned out not to be true, there would be little hope for a "scientific" approach to human engineering in the, then, foreseeable future.

In addition to the above-mentioned reasons for the development of a HOS, there was yet another reason. In those days, we were conducting some open-loop simulations of various missile and missile launch systems. In one conducted by Chuck Hutchins, it was discovered that operators in the laboratory simulation were getting very good scores on locking onto and launching a simulated missile at a simulated target. It was also discovered that the operators were waiting until minimum range to launch their simulated weapons. The simulated targets were capable of maneuvering, but the maneuvers were "canned" and had nothing to do with the maneuvering our pilots were doing. Further, the simulated targets never fired back, which might well account for the willingness of our pilots to wait until minimum range to release their missiles. Thus, the concept of a simulated human operator to be used as an intelligent adversary was also one of the original planned uses of a HOS (although, to date, HOS has never been used for this purpose).

In formulating the philosophy of how one could simulate a human operator's behavior, one major concern I had in 1967 was whether a human being could be considered to be a discrete or continous information processor. In those days, many people held to the concept that man was indeed a continuous processor. If this were true, it might be more appropriate to use an analog rather than a digital computer. However, by reanalyzing some data collected much earlier by John Senders, I came to the conclusion that even in a continuous tracking task, the human appeared to be sampling the available displayed information only about 13 times per second. Thus, man appeared, at least to me, not to be a continuous sampler, but a discrete one who could relatively easily be simulated with a digital computer.

Another major philosophical point was whether man should be considered to be a single- or multi-channel processor. This is more than a question of whether an operator can be responsible for carrying out more than one task at a time, for this he might appear to do even if he were a single-channel operator capable of very rapid interlacing among more than one task. The single-channel vs. multi-channel question really revolves on the issue of whether the operator can *simultaneously* be thinking about two different things. After much introspection (as well as considering the writings of various experts on this question) I chose essentially to conceive of man as being a single-channel processor who is capable of rapidly multiplexing among several tasks.

This, in turn, led to the concept in HOS of permitting the simulated operator to have many different procedures going on at the same time. In HOS, we call these the "active" procedures, while those which are not currently of concern to the operator are known as "inactive" ones. However, while many tasks may be "active," HOS only works on one at a time.

One of the earliest studies I did (long before HOS or the HOPROC language existed) was a relatively simple computer simulation to determine what would happen under various strategy algorithms for deciding which displays to pay attention to when the simulated operator was responsible for monitoring several different ones at the same time. These early studies led to the concepts of a MONITORING PROCEDURE for a display as well as the concepts of a procedure's "CRITICALITY" and the idea that criticality could dynamically change as a function of the disparity between a display's "desired position," its "allowable limits", and its "estimated position." These concepts have been retained in HOS since its beginning stages back in early 1968. Such algorithms provide the basis for the "adaptiveness" of the behavior exhibited by HOS.

Another very early consideration (which has changed very little ovar the years) was how to handle "short-term" memory of the simulated operator. The concept of "HAB" strength (which is discussed elsewhere) and the probability of successful recall of an item of information which had been recently absorbed was a concept which I adapted from Hull's and Thorndike's theories of learning. The concept of modeling short-term memory was felt to be necessary to determine how often the human would feel a necessity to update his current information about some parameter by actually looking at a display.

By 1968, the basic concepts of HOS which included micro-process handlers, adaptiveness algorithms, short-term memory, with the operator as a single-channel processor capable of rapid multiplexing among the "active" procedures had been formulated in detail as well as the earliest version of the HOPROC language by which the user would specify what it was that the simulated operator was expected to do. These concepts were reported in the proceedings of a two-day meeting jointly hosted by the Office of Naval Research and North American Aviation in Columbus, Ohio in November, 1968. It is surprising and somewhat rewarding to see how little the basic concepts formulated 10 years ago have changed during its development. It is also interesting to note that I and other participants at that meeting estimated that it might take 10 years to develop HOS.

By 1969, I was able to get some Independent Research (6.1) funds to pay for a programmer (Mr. Don Kennerly -- then a member of our HE branch) to start programming both the earliest versions of HOS and HAL (the HOPROC Assembler/Loader program which was to decipher the HOPROC statements for input to the HOS program). These earliest programs did not include all the specifications of HOS mentioned above and HAL was written in COBOL. More than anything else, they proved, at least to my own satisfaction, that it would be feasible to write a digital computer program for a full-blown HOS. It was also in 1969 that Bittner and I did the experiments mentioned above which also were very encouraging regarding the concept of the additivity of micro-process times.

In August of 1970, I was transferred to the Naval Air Development Center in Warminster, Pennsylvania and it was immediately obvious that a HOS would be even more valuable during early system development than during later test and evaluation phases of system design.

Paul Chatelier, who was at that time stationed at NADC, was in the throes of formulating CAFES which also dealt with computerized approaches to improving human factors engineering technology. (Later funding for HOS was formally included in the CAFES program element number, but for two years they stayed as separate development efforts.)

By December, 1970 , Analytics became interested in the HOS concept and submitted a proposal to work on its further development. Prior to that, I had discovered that although I had brought the HOS and HAL programs (written by Kennerly) with me, NADC did not have a version of COBOL which could compile the HAL program as it then existed. It was decided that it would be better to have all the future programs written in FORTRAN for subsequent ease in transferring them about the country.

The first contract to Analytics for work on HOS was let in 1971 and out of that effort what I might call HOS II and HAL II were developed. It is interesting to note that none of the original Analytics team which started with that project are any longer involved with the work Analytics has done in the past six years on the HOS project.

As more work was accomplished on HOS, it became obvious that various additional statements in the HOPROC lanugage would be desirable as well as a greater flexibility in how one could express various statements. For a while, these additions were added as patches to the program until it became obvious that it was time to go back and incorporate all these changes as well as some additional new concepts into the HAL and HOS programs. Thus, what has been available since late 1975 actually is what we might call HOS III and HAL III. Since that time, we have almost exclusively been involved in validity testing of HOS III and little or no additional development has taken place.

This does not mean to imply that HOS is considered to be in its final or ultimate stage of development, for there are many additional features which should and could be added to HOS. However, HOS III does represent what I consider to be a highly useful technique for the initial assessment of how well a trained operator will be able to perform his tasks in a specified crewstation under varying situational demands.

One concept which was definitely added to HOS in 1974 which was a rather marked departure from original plans for HOS, was the concept of simulating the system hardware and software as well as targets using the HOPROC language and the HAL and HOS programs. Originally, HOS was only to be the Human Operator Simulator and it was anticipated that it would be interfaced to hardware simulators in some fashion. It was found, however, that it would be extremely difficult to modify hardware simulators written by others so that HOS could easily interface with them. After much soul searching, it was decided to expand the HOPROC language, HAL and HOS, to include the ability to simulate hardware as well as the human components. These changes were also incorporated and indeed necessitated the rewriting for HOS III and HAL III.

The concept of a HODAC (Human Operator Data Analyzer/Collator) program to analyze the human operator data emminating from a HOS run was included in the very early stages of HOS planning. The first HODAC, however, was not available until 1974. It has proven to be less useful than I originally thought it might, but this may be due, in part, to the fact that we have to date been most interested in seeing if HOS behaves like real operators in systems which have already actually been built (i.e., our validating studies) rather than in systems which are actually under development. It may be that many of the routines available in HODAC will turn out to be very useful in deciding potential changes to procedures and crewstation design when we try HOS on a developing systems and we determine that unless something is changed, it will be impossible for the operator to successfully do all his allocated functions.

While HAL III and HOS III both now contain the additional capability for simulating hardware and target systems, there is no automatic logging of their behavior as there is with the simulated human behavior. In part, this is due to the fact that HOS does not contain a "general purpose" hardware system model which is made system specific by the hardware procedures and hardware functions. Lacking an overall scheme for a general hardware system means that hardware systems are not automatically reducible to a specified number of micro-processors which can then be automatically logged out whenever they are used. This necessitates some amount of cleverness on the part of the HOS user to either log out and/or accumulate data of interest to overall system performance.

Earlier, I mentioned that HOS should not be considered to be fully developed. Areas where HOS might be expanded include the addition of a "fatigue" model, the capability to pick up and move objects from one place to another, the capability to walk (or run) from one place to another, the ability to talk to another operator, the capability to perform visual target recognition in a complex visual field, etc. I am convinced that each of the above concepts can be added to HOS and I have, at least, rudimentary models or schemes for handling all of the above concepts as well as several others. With the rather successful validation studies which have been conducted on HOS III, it is probably now time to start the development of HOS IV and HOS V which would be versions to include one or more of the above concepts.

Finally, I would be remiss if I did not discuss the concept of operator error as it is treated in HOS. More than any other single item, the way operator error is treated in HOS has been criticized. With a few exceptions (which are discussed elsewhere), HOS does not make errors. Some people claim that this is unrealistic, but I maintain that as long as the human operator is given the requisite amount of time to do a task, then he does not make errors. He may not finish all the tasks we would like him to do, and he may not do them as well as we would like him to do them, but as long as he works at a reasonable pace, he will not make an error. The fact that he doesn't get all his tasks accomplished when he works at a reasonable pace merely indicates that we allocated too many tasks to him. Thus, if HOS indicates the simulated operator spends too little time on a given task, we may either assign a higher criticality to that task and rerun the simulation to see if this alleviates the problem, or we may reduce the number of tasks which were originally allocated to the operator to see if this solves the problem.

I am certain that real systems do exist in which operators make mistakes. On the other hand, this is a clear indication that we have, in those systems, asked the operator to do too many and/or too complicated tasks and therefore those systems are not properly human engineered by definition, since successful performance of the tasks are not within the capabilities of the operator. What HOS does is essentially to "instruct" the operator not to attempt to work at a pace at which errors and mistakes will occur because we hold to the concept that *errors are the result of being under time stress* and that error-free performance can be maintained provided the operator does not attempt to do too many things in too short of a time period. The human errors that are observed in existing systems are the result of real operators attempting, unsuccessfully, to perform at a higher level than their capabilities permit.

In closing this brief historical perspective, I should also mention that working on the development of HOS has been both fun and exciting. The associations I have formed with the many people who have participated in this development program has been very rewarding professionally over the years. Finally, working on HOS has also often been a humbling proposition as we have discovered how little we actually know about how humans behave.

REFERENCES

Analytics. 1975. *Introduction and Overview.* The Human Operator Simulator. Vol. 1. Technical Report 1117-I. Willow Grove, Pennsylvania.

---. 1975. *HOS Users' Guide.* The Human Operator Simulator. Vol. 2. Technical Report 1181-A. Willow Grove, Pennsylvania.

---. 1976. *HAL Programmers Guide.* The Human Operator Simulator. Vol. 3. Technical Report 1181-C. Willow Grove, Pennsylvania.

---. 1974. *HOS Programmers Guide.* The Human Operator Simulator. Vol. 4. Technical Report 1117-B. Willow Grove, Pennsylvania.

---. 1974. *HODAC Programmers Guide.* The Human Operator Simulator. Vol. 5. Technical Report 1117-A. Willow Grove, Pennsylvania.

---. 1975. *HOS Simulation Descriptions.* The Human Operator Simulator. Vol. 6. Technical Report 1181-B. Willow Grove, Pennsylvania.

---. 1976. *LAMPS Air Tactical Officer Simulation.* The Human Operator Simulator. Vol. 7. Technical Report 1200. Willow Grove, Pennsylvania.

---. 1977. *Applications to Assessment of Operator Loading.* The Human Operator Simulator. Vol. 8. Technical Report 1233-A. Willow Grove, Pennsylvania.

---. 1978. *P-3C SS-3 Operator Station Cost Effectiveness Evaluation.* The Human Operator Simulator. Vol. 10. Technical Report 1289. Willow Grove, Pennsylvania.

Bradley, J.V., and Wallis, R.A. 1959. Spacing of pushbutton on-off controls. *Engineering and Industrial Psychology.* 1:107-119.

Bradley, J.V., and Wallis, R.A. 1960. Spacing of toggle switch on-off controls. *Engineering and Industrial Psychology.* 1:8-19.

Bush, R.R., and Mosteller, F. 1955. *Stochastic models for learning.* New York: John Wiley & Sons, Inc.

Chapanis, A., and Kinkade, R.G. Design of controls. In Van Cott, H.P., and Kinkade, R.G. 1972. *Human engineering guide to equipment design* (revised edition). U.S. Government Printing Office.

Cohen, V.V.R. 1971. Short-term memory for quantitative information from three kinds of visual displays. Human Performance Center Technical Report No. 28, University of Michigan, Ann Arbor, Michigan.

Crossman, E.R.F.W. 1960. The information capacity of the human motor system in pursuit tracking. *Quarterly Journal of Experimental Psychology.* 12:1-16.

Dansereau, D.F., and Gregg, L.W. 1966. An information processing analysis of mental multiplication. *Psychonomic Science.* 6:71-72.

Davies, B.T., and Watts, J.W. 1969. Preliminary investigation of movement time between brake and acceleration pedals in automobiles. *Human Factors.* 11:257-272.

Dodge, R., and Cline, T.S. 1901. The angle velocity of eye movements. *Psychology Review.* 8:145-157.

Drury, C.G. 1975. Application of Fitts' Law to foot-pedal design. *Human Factors.* 17:368-373.

Fitts, P.M. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology.* 47:381-391.

Fitts, P.M., and Peterson, J.R. 1964. Information capacity of discrete motor responses. *Journal of Experimental Psychology.* 67:103-112.

Goldbeck, R.A., and Charlet, J.D. 1974. Task parameters for predicting panel layout design and operator performance. Philco-Ford Technical Report WLD-TR5480. Philco-Ford Corporation, WDL Division, Palo Alto, California.

Karger, D.W., and Bayha, F.H. 1966. *Engineered work management.* New York: Industrial Press, Inc.

Macpherson, D.H., and Siegel, A.I. 1967. Verification of a digital technique for sonar operation simulation. APS Technical Report, Applied Psychological Services, Science Center, Wayne, Pennsylvania.

Melton, A.W., Crowder, R.G., and Wolff, D. Short-term memory for individual items with varying numbers of elements. Cited in Melton, A.W. 1963. Implications of short-term memory for a general theory of memory. *Journal of Verbal Learning and Verbal Behavior.* 2:1-21.

Murdock, B.B. 1961. The retention of individual items. *Journal of Experimental Psychology.* 62:618-625.

Naval Air Development Center. 1974. *LAMPS MK-III ((E-2) Short-range) operator's manual for pilot and airborne tactical officer.* NADC 5013-57. Warminster, Pennsylvania.

Peterson, L.R., and Peterson, M.J. 1959. Short-term memory of individual verbal items. *Journal of Experimental Psychology.* 58:193-198.

Pollatsek, A.W. 1969. Rehearsal, interference, and spacing of practice in short-term memory. Human Performance Center Technical Report No. 16, University of Michigan, Ann Arbor, Michigan.

Quick, J.H., Duncan, J.H., Jr., and Malcolm, J.A., Jr. 1962. *Work-factor time standards*. New York: McGraw Hill Book Co.

Restle, F. 1970. Speed of adding and comparing numbers. *Journal of Experimental Psychology*. 83:274-278.

Seibel, R. Data entry devices and procedures. In Van Cott, H.P., and Kinkade, R.G. 1972. *Human Engineering guide to equipment design* (revised edition). U.S. Government Printing Office.

Shannon, C., and Weaver, W. 1949. *The mathematical theory of communication*. Urbana: University of Illinois Press.

Thomas, H.B.G. 1963. Communication theory and the constellation hypothesis of calculation. *Quarterly Journal of Experimental Psychology*. 15:173-191.

Topmiller, D.A., and Sharp, E.D. 1965. Effects of visual fixation and uncertainty on control panel layout. Technical Report AMRL-TR-65-149, Aerospace Medical Research Laboratories, Wright-Patterson Air Force Base, Ohio.

Waugh, N.C. 1969. The effects of recency and repetition on recall latencies. *Acta Psychologica*. 30:115-125.

Welford, A.T. 1960. The measurement of sensory-motor performance survey and reappraisal of twelve years' progress. *Ergonomics*. 3:189-230.

DATE

ILMED

- 8